Approved for public release; distribution is unlimited

Parallelization of the Air Force Space Command (AFSPACECOM) Satellite Motion Models

by

Sara R. Ostrom
Lieutenant, United States Navy
B. S., University of Arizona, 1986

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
March 1993

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (If applicable) MA	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)

PARALLELIZATION OF THE AIR FORCE SPACE COMMAND (AFSPACECOM) SATELLITE MOTION MODELS

12. PERSONAL AUTHOR(S)
Ostrom, Sara R.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1993 March	

16 SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government

17. COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Parallel Processing, Satellite Motion Model, Hypercube

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The Air Force Space Command (AFSPACECOM) uses an analytic satellite motion model based on the Brouwer theory, referred to as SGP4, for near-earth objects, and another satellite motion model based on the work of Hujsak, referred to as SDP4, for deep-space objects. These models assist in tracking over 7000 objects around the Earth each day. Also, the original satellite motion model used by the AFSPACECOM based on the work of Kozai and Brouwer, known as the Simplified General Perturbations model or SGP, is an analytic model still being used at several sensor sites. Based on the increasing number of space objects that require tracking and the desire for increased accuracy, there is a growing need to reduce computation time in implementing satellite motion models. Parallel computing offers one method to achieve this objective. This thesis investigates the parallel computing potential of SGP, SGP4, and SDP4 using the Intel iPSC/2 hypercube multicomputer. This thesis chooses a parallel algorithm and applies it to SGP, SGP4, and SDP4 for implementation on the hypercube, and reports on the potential reduction in computer time by first considering only the calculational portion of the algorithm and then including the effects of the reading and writing portions. A diskette containing the Fortran software developed is available upon request from bneta@moon.math.nps.navy.mil.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS		
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. Beny Neta, Prof. Don Danielson	22b TELEPHONE (Include Area Code) (408)656-2235, (408)656-2622	22c. OFFICE SYMBOL MA/Nd, MA/Dd

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603 — Unclassified

ABSTRACT

The Air Force Space Command (AFSPACECOM) uses an analytic satellite motion model based on the Brouwer theory, referred to as SGP4, for near-earth objects, and another satellite motion model based on the work of Hujsak, referred to as SDP4, for deep-space objects. These models assist in tracking over 7000 objects around the Earth each day. Also, the original satellite motion model used by the AFSPACECOM based on the work of Kozai and Brouwer, known as the Simplified General Perturbations model or SGP, is an analytic model still being used at several sensor sites. Based on the increasing number of space objects that require tracking and the desire for increased accuracy, there is a growing need to reduce computation time in implementing satellite motion models. Parallel computing offers one method to achieve this objective. This thesis investigates the parallel computing potential of SGP, SGP4, and SDP4 using the Intel iPSC/2 hypercube multicomputer. This thesis chooses a parallel algorithm and applies it to SGP, SGP4, and SDP4 for implementation on the hypercube, and reports on the potential reduction in computer time by first considering only the calculational portion of the algorithm and then including the effects of the reading and writing portions. A diskette containing the Fortran software developed is available upon request from bneta@moon.math.nps.navy.mil.

TABLE OF CONTENTS

ACKNOWLEDGMENT

To my thesis advisors, Professor Beny Neta and Professor Don Danielson, I greatly appreciate the patience, guidance, and professional teamwork they provided from the beginning to completion of this thesis. I am also thankful for their belief and confidence in my abilities, which motivated me to accomplish more than I originally set out to do.

To my information source and contact person at the AFSPACECOM, Denise Kaya, I am most grateful for her cooperation in providing the software, technical reports, and technical advice necessary to do my research.

To my husband, Keith, I sincerely thank for his unwavering support, love, and patience. He took on many extra responsibilities to give me the much needed time to perform my thesis work.

To my daughter, Jessica, I appreciate her companionship at many long hours spent at the computer.

Finally, I am most thankful to God. Without whom this thesis would not have been possible.

I. INTRODUCTION

The first satellite motion model used by the Air Force Space Command (AFSPACECOM) is known as the Simplified General Perturbations (SGP) satellite motion model. SGP is an analytic model based on the work of Kozai (1959) and Brouwer (1959), made operational by Hilton and Kuhlman (1966). It is implemented by a Fortran subroutine, SGP, and is used to track near-Earth satellites (period less than 225 minutes) having orbits with low eccentricities. By 1976 a second satellite motion model for near-Earth satellites was implemented by a Fortran subroutine, SGP4, replacing SGP as the operational satellite motion model used by the AFSPACECOM. SGP4 is an analytic satellite motion model based on the theory developed by Lane and Cranford (1969) which uses the solution of Brouwer (1959). To meet the needs for deep-space satellites another model, SDP4, was made operational. SDP4 is an extension of SGP4 where the deep-space theory is derived from Hujsak (1979). Currently, SGP4 refers to one Fortran subroutine that has combined the former separate subroutines SGP4 and SDP4, and is what the AFSPACECOM now uses to generate the North American Aerospace Defense and Command (NORAD) element sets.

The AFSPACECOM currently tracks approximately 7000 space objects on a daily basis. As space technology progresses and as the dependence upon applications in space increases, the number of space objects that require tracking will continue to grow. The future use of operational satellite motion models used by the AFSPACECOM need to involve shorter computational time to obtain near real time orbit predictions. Also, if the AFSPACECOM chooses to develop models with greater accuracy, these models will

undoubtedly require even more computational time. Such accurate models can be used to predict orbits of smaller objects, increasing the number of objects by ten fold.

Computational time in implementing a satellite motion model is highly dependent upon the computer configuration used. Parallel computing has the potential to significantly decrease computational time over serial computing currently used by the AFSPACECOM. Use of parallel computers has already proven to be beneficial in reducing computation time in many other applications. Once a decision is made to convert to parallel computing, future satellite motion models can be developed in a parallel architecture format. Ideally, these models made operational with the use of parallel computers will result in computationally efficient programs that yield highly accurate results.

This thesis investigates the parallel computing potential of the AFSPACECOM's first operational satellite motion model, SGP, and the current satellite motion models, SGP4 and SDP4, using the Intel iPSC/2 hypercube multicomputer. The SGP model was included since it is still used at several surveillance sensor sites. The following chapter provides a description of the SGP model and outlines the algorithm used by the Fortran subroutine, SGP. Chapter III gives a general description of the SGP4 and SDP4 models and how they compare with the SGP model. Chapter IV presents an overview of parallel processing and discusses the methods to parallelize a serial algorithm to be applied to the hypercube. In Chapter V, the parallelization of SGP, SGP4, and SDP4 are presented with their respective success in reducing computation time. Here the effects of reading and writing to and from the disk are not included. A comparison between the parallelized versions of SGP, SGP4, SDP4 and PPT2 is also included. PPT2 is an analytic satellite motion model used by the Naval Space Surveillance Center (NAVSPASUR), and a parallelized version was created by CPT Warren E. Phipps (1992) at the Naval Postgraduate School using the same multicomputer, Intel iPSC/2

hypercube. Chapter VI further investigates the parallel computing potential of SGP4 and SDP4 by considering the effects of increased computation time and the effect of reading and writing to and from the disk. The last chapter of this thesis provides conclusions and suggestions for future research.

II. SIMPLIFIED GENERAL PERTURBATIONS (SGP) MODEL

A. OVERVIEW

The Simplified General Perturbations model is the original model used by the AFSPACECOM to track artificial satellites orbiting the earth, implemented by the subroutine SGP. Given pseudo elements generated by the AFSPACECOM, SGP users can predict the state vector (position and velocity) at a future time. This model considers perturbing accelerations due to the oblateness of the Earth, asymmetry of the Earth's mass about the equatorial plane, and atmospheric drag. This model does not include perturbation effects due to longitudinal variation in the Earth's gravitational potential, or due to other celestial bodies such as the moon or the sun.

Satellite motion models can be classified according to how the equations of motion are solved. The two general classifications are known as *general perturbations* and *special perturbations*. General perturbation models involve solving the equations of motion analytically. As more perturbations are included in a satellite motion model to increase accuracy, analytical solutions become increasingly complicated if not impossible to solve. On the other hand, it is always possible to use special perturbations. Special perturbation models involve solving the equations of motion through numerical integration. Although special perturbation models are generally more accurate and less complicated than general perturbation models, they are computationally expensive. A third model type known as a semi-analytic model combines the general and special perturbation techniques in an attempt to balance the advantages and disadvantages of the two techniques, resulting in a highly accurate and computationally efficient model. The SGP model is an example of a general perturbations model.

Satellite motion models can also be classified according to how the variation in the satellite's motion is represented. The two general classifications are known as *variation of elements* and *variation of coordinates*. Variation of elements identifies variations in terms of changes in the osculating elements with respect to time. Variation of coordinates involves choosing a coordinate system and describing the changes in position and velocity with respect to time in that coordinate system. The SGP model uses variation of elements and describes the changes in the classical orbital elements with respect to time.

B. THEORY

The Simplified General Perturbations (SGP) model is an analytic model developed by Hilton and Kuhlman (1966). SGP's gravitational submodel is a simplification of the work done by Kozai (1959) and Brouwer (1959). The atmospheric drag submodel describes the mean motion as a linear function of time.

Before proceeding with the theoretical discussion of these models, a brief explanation of how SGP defines satellite motion is needed. The variation in the motion due to the perturbation effects is described by the changes in the classical orbital elements with respect to time. The six classical elements are n, e, i, Ω, ω, and M (see Figure 2.1), where

n = *mean motion*
e = *eccentricity*
i = *inclination of the orbital plane to the equator*
Ω = *right ascension of the ascending node*
ω = *argument of perigee*
M = *mean anomaly at epoch.*

The position and velocity vectors can be obtained from these six orbital elements.

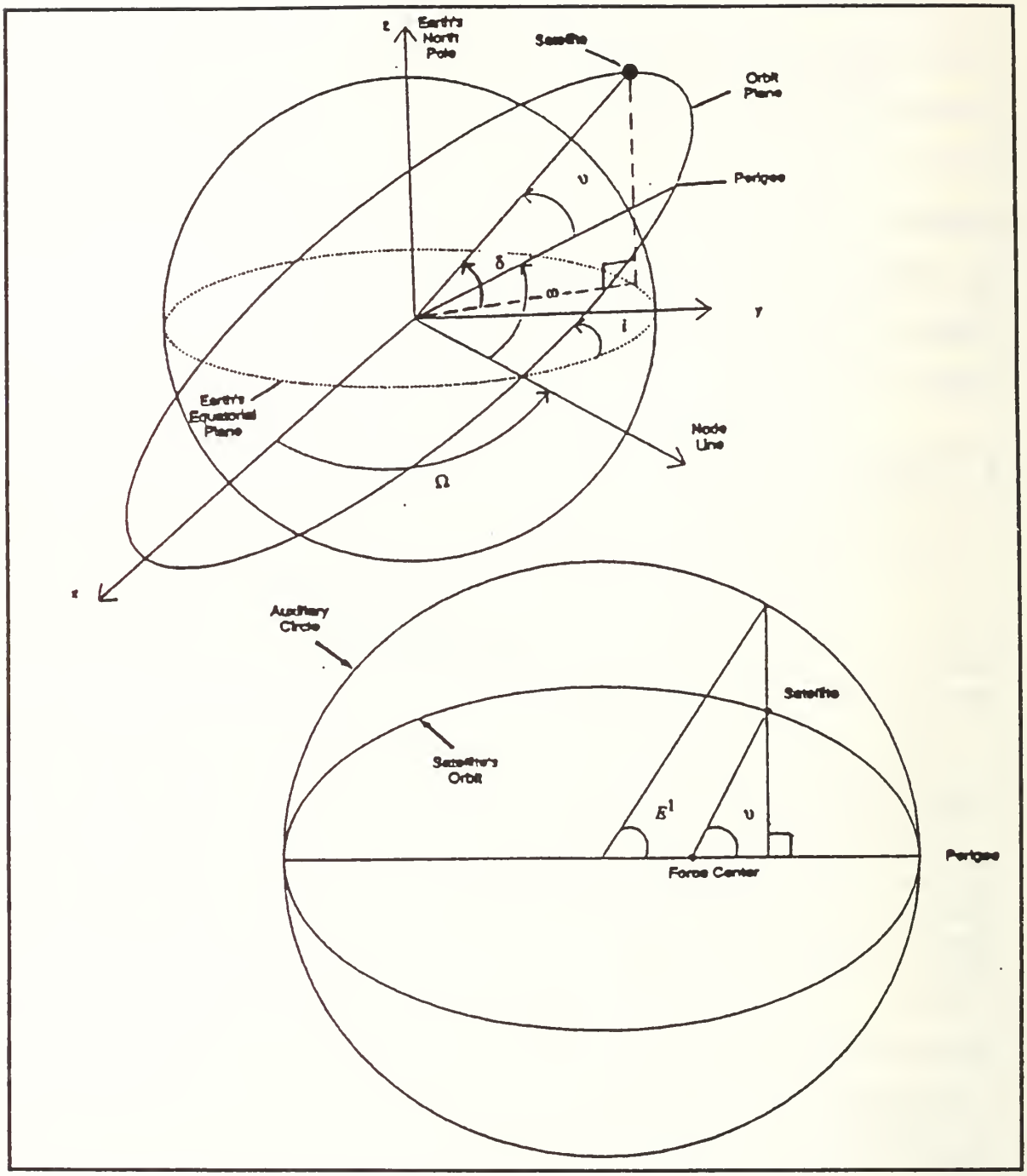

Figure 2.1 Classical Orbital Elements

---

[1] E represents the eccentric anomaly and is related to the mean anomaly by,

$$E - e \sin E = M.$$

1. Kozai's Gravitational Model

Here a simplification of Kozai's model is presented (Roy, 1988, pp. 312-320). Kozai's model is more involved than the classical Keplerian model for idealized satellite motion. In the classical model the Earth's potential at an external point, a distance r from its center, is expressed by

$$U = \frac{\mu}{r} \tag{2.1}$$

where μ is the gravitational parameter, equal to the product of the Earth's mass and the gravitational constant G. Thus, a satellite of mass m at a distance r possesses a potential energy equal to

$$-mU = -m\frac{\mu}{r}.$$

Equation 2.1 represents the potential for a perfectly spherical Earth. In Kozai's model, the Earth is considered to be a planet that is symmetric about the spin axis and asymmetric about the equatorial plane . Thus, the Earth's potential may be written as

$$U = \frac{\mu}{r}\left[1 - \sum_{n=2}^{\infty} J_n \left(\frac{R}{r}\right)^n P_n(\sin\delta)\right] \tag{2.2}$$

where J_n are constants, R is the Earth's equatorial radius, δ is the declination, and $P_n(\sin\delta)$ is the Legendre Polynomial of order n in $\sin\delta$.

The Earth's potential can be written in terms of a disturbing function F

$$U = U_0 + F$$

where U_0 represents the classical potential. Thus the disturbing function can be written as

$$F = U - U_0 = U - \frac{\mu}{r} = -\frac{\mu}{r}\sum_{n=2}^{\infty} J_n \left(\frac{R}{r}\right)^n P_n(\sin\delta). \tag{2.3}$$

Now, J_2 is of the order 10^{-3}, and $J_3, J_4, \ldots$ are of order 10^{-6} or less. Since $J_4, J_5, \ldots$ are relatively insignificant and the $\left(\frac{R}{r}\right)^n$ factor decreases as n increases, further analysis will only include J_2 and J_3, the second and third harmonics, in agreement with the SGP model. Thus, the disturbing function can be written as

$$F = -\frac{\mu}{r}\left[J_2\left(\frac{R}{r}\right)^2\left(\frac{3}{2}\sin^2\delta - \frac{1}{2}\right) + J_3\left(\frac{R}{r}\right)^3\left(\frac{5}{2}\sin^2\delta - \frac{3}{2}\right)\sin\delta\right]. \tag{2.4}$$

Substituting the known relations,

$$r = \frac{a\left(1-e^2\right)}{1+e\cos\upsilon}$$

and

$$\sin\delta = \sin i \sin(\upsilon + \omega),$$

where υ is the true anomaly and δ is the declination , as illustrated in Figure 2.1, F becomes

$$\begin{aligned} F = \mu\Bigg\{&\frac{3}{2}\frac{J_2R^2}{a^3}\left(\frac{a}{r}\right)^3\left[\frac{1}{3} - \frac{1}{2}\sin^2 i + \frac{1}{2}\sin^2 i\cos 2(\upsilon+\omega)\right] \\ &-\frac{J_3R^3}{a^4}\left(\frac{a}{r}\right)^4\left[\left(\frac{15}{8}\sin^2 i - \frac{3}{2}\right)\sin(\upsilon+\omega) - \frac{5}{8}\sin^2 i\sin 3(\upsilon+\omega))\right]\sin i\Bigg\}. \end{aligned} \tag{2.5}$$

Next, the disturbing function is separated into first-order secular, second-order secular, long-period, and short-period terms. These parts are referred to as F_1, F_2, F_3, and F_4 respectively. Now, F is periodic with respect to the true anomaly υ and the radial distance r. The true anomaly υ and the mean anomaly M are related by

$$\frac{d\upsilon}{dM} = \frac{a^2}{r^2}\left(1-e^2\right)^{\frac{1}{2}}. \tag{2.6}$$

Since the quantities $\frac{r}{a}$ and υ are functions of e and M only, F is periodic with respect to M. Terms in F depending on M and not on ω are short-period. Terms in F depending on ω but not on M are long-period. The resulting terms depending neither on M nor on ω are secular. Thus, the terms of F are sorted in the following way:

F_1: These terms are obtained by averaging with respect to M those parts of the first portion (i.e., J_2 part) of F that are independent of M and ω.

F_2: These terms would be obtained by averaging with respect to M those parts of the second portion (i.e., J_3 part) of F that are independent of M and ω, but since there are no such terms F_2 is zero.

F_3: These terms are obtained by taking the mean value of the second portion of F with respect to M.

F_4: These terms are obtained by subtracting the first-order secular terms from the first portion of F.

The mean values are derived by

$$\overline{Q} = \frac{1}{2\pi}\int_0^{2\pi} Q\,dM$$

where Q represents any term in F that is being averaged, and the equations,

$$\begin{aligned}
&\overline{\left(\frac{a}{r}\right)^3} = \frac{1}{2\pi}\int_0^{2\pi}\left(\frac{a}{r}\right)^3 dM = \left(1-e^2\right)^{-\frac{3}{2}} \\
&\overline{\left(\frac{a}{r}\right)^3 \sin 2\upsilon} = \overline{\left(\frac{a}{r}\right)^3 \cos 2\upsilon} = 0 \\
&\overline{\left(\frac{a}{r}\right)^4 \cos\upsilon} = e\left(1-e^2\right)^{-\frac{5}{2}} \\
&\overline{\left(\frac{a}{r}\right)^4 \sin\upsilon} = \overline{\left(\frac{a}{r}\right)^4 \cos 3\upsilon} = \overline{\left(\frac{a}{r}\right)^4 \sin 3\upsilon} = 0,
\end{aligned} \tag{2.7}$$

by Tisserand (1889) are used to obtain F_1, F_2, F_3, and F_4. The resulting equations are

$$
\begin{aligned}
F_1 &= \frac{3}{2}\frac{\mu J_2 R^2}{a^3}\left(\frac{1}{3}-\frac{1}{2}\sin^2 i\right)\left(1-e^2\right)^{-\frac{3}{2}} \\
F_2 &= 0 \\
F_3 &= \frac{3}{2}\frac{\mu J_2 R^3}{a^4}\sin i\left(1-\frac{5}{4}\sin^2 i\right)e\left(1-e^2\right)^{-\frac{5}{2}}\sin\omega \\
F_4 &= \frac{3}{2}\frac{\mu J_2 R^2}{a^3}\left(\frac{a}{r}\right)^3\left\{\left(\frac{1}{3}-\frac{1}{2}\sin^2 i\right)\left[1-\left(\frac{r}{a}\right)^3\left(1-e^2\right)^{-\frac{3}{2}}\right]\right. \\
&\quad \left.+\frac{1}{2}\sin^2 i\cos 2(\upsilon+\omega)\right\}
\end{aligned}
\tag{2.8}
$$

2. SGP's Gravitational Model

This model follows directly from the theory discussed in part one. It will be presented based on Project Space Track Report No. 3 (Hoots and Roehrich, 1980), maintaining a parallel structure between the equations and the computer code.

a. Calculating Initial Constants

Predictions are made by first calculating the constants below:

(1) Semi-major axis. This is expressed by

$$
a_1 = \left(\frac{k_e}{n_o}\right)^{\frac{2}{3}} \tag{2.9}
$$

where

a_1 = *semimajor axis*

$k_e = \sqrt{\mu}$

n_o = *SGP type "mean" mean motion at epoch.*

(2) Delta1 (δ_1). This is an intermediate calculation in modifying the semi-major axis to include perturbation effects due to the oblateness of the Earth. Delta1 is expressed by

$$
\delta_1 = \frac{3}{4}J_2\frac{{a_E}^2}{{a_1}^2}\frac{\left(3\cos^2 i_o - 1\right)}{\left(1-{e_o}^2\right)^{\frac{3}{2}}} \tag{2.10}
$$

where

$$a_E = equatorial\ radius\ of\ Earth$$
$$i_o = "mean"\ inclination\ at\ epoch$$
$$e_o = "mean"\ eccentricity\ at\ epoch.$$

Delta1 will be revealed in the next part.

(3) Modified Semi-major Axis (a_o). This is expressed by

$$a_o = a_1\left[1-\frac{1}{3}\delta_1-\delta_1{}^2-\frac{134}{81}\delta_1{}^3\right]. \tag{2.11}$$

Equation 2.11 is derived in the following manner:

By substituting F_1 for F in the differential equation (Roy, 1988, p. 316)

$$\frac{dM_o}{dt} = n_o - \frac{1-e_o{}^2}{n_o a_o{}^2 e_o}\frac{\partial F}{\partial e_o} - \frac{2}{n_o a_o}\frac{\partial F}{\partial a}$$

the mean "mean" anomaly at epoch M_o is obtained,

$$M_o = \tilde{M}_o + n_o t$$

where $\tilde{M}_o$ is the unperturbed mean anomaly, with

$$n_o = \tilde{n}_o\left[1+\frac{3}{4}J_2\frac{a_E{}^2}{a_o{}^2\left(1-e_o{}^2\right)^{\frac{3}{2}}}\left(3\cos^2 i_o - 1\right)\right]$$

where $\tilde{n}_o$ is the unperturbed mean motion. For convenience let

$$D = \frac{3}{4}J_2\frac{a_E{}^2}{\left(1-e_o{}^2\right)^{\frac{3}{2}}}\left(3\cos^2 i_o - 1\right).$$

Now, a_o is chosen by convention to be

$$a_o = \tilde{a}_o\left(1-Da_o{}^{-2}\right)$$

where $\tilde{a}_o$ is the unperturbed semi-major axis. Thus, using the relation

$$\tilde{a}_o{}^3\tilde{n}_o{}^2 = k_e{}^2$$

leads to the equation

$$a_o^{\,3} n_o^{\,2} = k_e^{\,2}\left(1 - Da_o^{\,-2}\right)^3\left(1 + Da_o^{\,-2}\right)^2.$$

Therefore,

$$a_o = \left(\frac{k_e^{\,2}}{n_o^{\,2}}\right)^{\frac{1}{3}}\left(1 - Da_o^{\,-2}\right)\left(1 + Da_o^{\,-2}\right)^{\frac{2}{3}} = a_1\left(1 - Da_o^{\,-2}\right)\left(1 + Da_o^{\,-2}\right)^{\frac{2}{3}}. \tag{2.12}$$

Equation 2.12 defines a_o implicitly. This equation is solved iteratively by using Picard's method, where a_1 (see Equation 2.9) is used as the initial approximation (Office of Astrodynamic Applications HQ Fourteenth Aerospace Force, 1974, pp. 1-6), resulting in Equation 2.11. It should be noted that in practice, n_o is obtained through a differential correction process on a separate computer code (i.e., the main program DRIVER reads this quantity as part of the NORAD 2-line element set), and is accurate to the third order (in J_2), thus a_o is calculated to the same accuracy.

(4) Semilatus Rectum (p_o). This is expressed by

$$p_o = a_o\left(1 - e_o^{\,2}\right) \tag{2.13}$$

The semilatus rectum is the length of the chord from the force center to the ellipse, parallel to the minor axis bb', as illustrated in Figure 2.2.

(5) Radius at Perigee (q_o). This is expressed by (see Figure 2.2)

$$q_o = a_0\left(1 - e_o\right). \tag{2.14}$$

(6) Mean "Mean" Longitude at Epoch (L_o). This is the sum of the "mean" mean anomaly at epoch (M_o), the "mean" argument of perigee at epoch (ω_o), and the "mean" longitude of ascending node at epoch. Thus, L_o can be written as

$$L_o = M_o + \omega_o + \Omega_o. \tag{2.15}$$

(7) Change in the "mean" longitude of ascending node with respect to time $\left(\frac{d\Omega}{dt}\right)$. The equation for $\frac{d\Omega}{dt}$ can be derived from the relation (Roy, 1988, p.316)

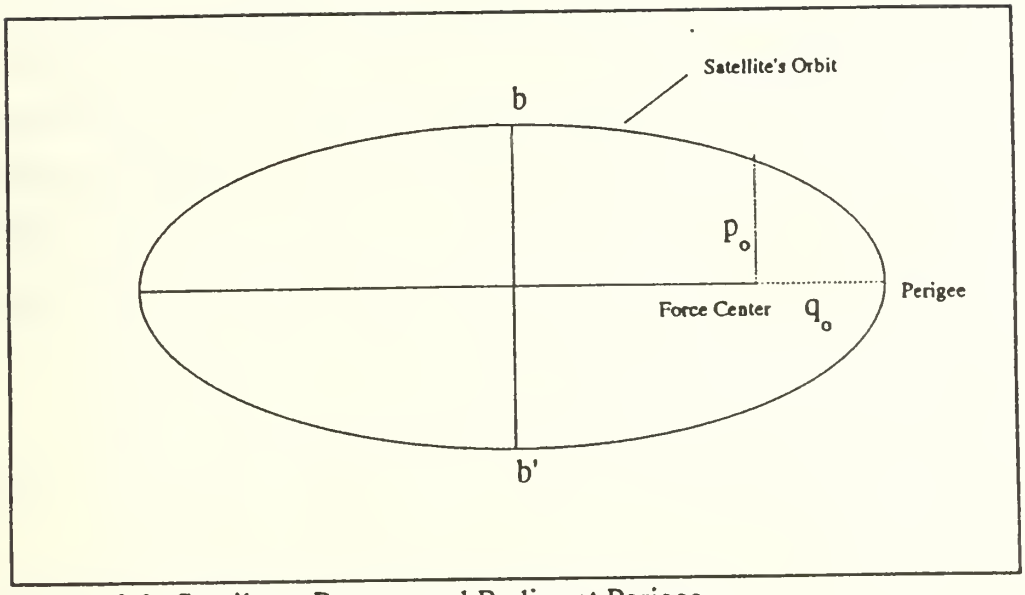

Figure 2.2 Semilatus Rectum and Radius at Perigee

$$\frac{d\Omega}{dt} = \frac{1}{n_o a_o^{\,2}\sqrt{1-e_o^{\,2}}\sin i_o}\frac{\partial F}{\partial i_o} \tag{2.16}$$

by substituting F_1 from Equation 2.8 for F in Equation 2.16 yielding

$$\frac{d\Omega}{dt} = -\frac{3}{2}J_2\frac{a_E^{\,2}}{p_o^{\,2}}n_o\cos i_o. \tag{2.17}$$

(8) Change in the "mean" argument of perigee with respect to time $\left(\frac{d\omega}{dt}\right)$.

The equation for $\frac{d\omega}{dt}$ can be derived from the relation (Roy, 1988, p.316)

$$\frac{d\omega}{dt} = -\frac{\cos i_o}{n_o a_o^{\,2}\sqrt{1-e_o^{\,2}}\sin i_o}\frac{\partial F}{\partial i_o} + \frac{\sqrt{1-e_o^{\,2}}}{n_o a_o^{\,2} e_o}\frac{\partial F}{\partial e_o}$$

by substituting F_1 from Equation 2.8 for F yielding

$$\frac{d\omega}{dt} = \frac{3}{4}J_2\frac{a_E^{\,2}}{p_o^{\,2}}n_o\left(5\cos^2 i_o - 1\right). \tag{2.18}$$

b. Secular Effects of Atmospheric Drag and Gravitation

The atmospheric drag is taken to effect mean motion linearly with time, resulting in a quadratic variation of mean anomaly with time. The drag effect on eccentricity is modeled based on the assumption that perigee height remains constant. The following analysis includes the secular effects of atmospheric drag and gravitation:

(1) Modified semi-major axis (a) to include atmospheric drag. This is expressed by

$$a = a_o \left(\frac{n_o}{n_o + 2\left(\frac{\dot{n}_o}{2}\right)(t - t_o)} \right)^{\frac{2}{3}} \tag{2.19}$$

where $t - t_o$ is the time since epoch. Equation 2.19 is derived from equation 2.16 which gives

$$a = a_o \left(\frac{n_o}{n} \right)^{\frac{2}{3}}$$

where n is approximated by a Taylor series expansion to the first order,

$$n = n_o + 2\left(\frac{\dot{n}_o}{2}\right)(t - t_o).$$

It should be noted that the term $\frac{\dot{n}_o}{2}$ is generated on a separate computer code which uses a differential corrector. The main program DRIVER reads this quantity as part of the NORAD 2-line element set.

(2) Modified eccentricity (e). This is expressed by

$$e = \begin{Bmatrix} 1 - \frac{q_o}{a}, & for\ a > q_o \\ 10^{-6}, & for\ a \le q_o \end{Bmatrix}. \tag{2.20}$$

Since the perigee radius is assumed constant, the resulting secular correction applied to the eccentricity may produce negative values for that element on near-circular orbits (Schumacher, 1991, pp. 106-107). A negative eccentricity is undesirable operationally. Thus, SGP arbitrarily resets these negative eccentricities to a small positive value $\left(10^{-6}\right)$.

(3) Modified semilatus rectum. This is expressed by

$$p = a\left(1 - e^2\right). \tag{2.21}$$

(4) Modified "mean" longitude of ascending node $\left(\Omega_{S_o}\right)$. This is expressed by

$$\Omega_{S_o} = \Omega_o + \frac{d\Omega}{dt}\left(t - t_o\right). \tag{2.22}$$

(5) Modified "mean" argument of perigee $\left(\omega_{S_o}\right)$. This is expressed by

$$\omega_{S_o} = \omega_o + \frac{d\omega}{dt}\left(t - t_o\right). \tag{2.23}$$

(6) Modified mean "mean" longitude $\left(L_S\right)$. This is expressed by

$$L_S = L_o + \left(n_o + \frac{d\omega}{dt} + \frac{d\Omega}{dt}\right)\left(t - t_o\right) + \frac{\dot{n}_o}{2}\left(t - t_o\right)^2. \tag{2.24}$$

Equation 2.24 holds since

$$\frac{dM_{S_o}}{d\left(t - t_o\right)} = n,$$

where, M_{S_o} may be approximated by

$$M_{S_o} = n_o t_o + n_o\left(t - t_o\right) + \frac{1}{2}\dot{n}_o\left(t - t_o\right)^2 = M_o + n_o\left(t - t_o\right) + \frac{1}{2}\dot{n}_o\left(t - t_o\right)^2. \tag{2.25}$$

Therefore

$$L_S = M_{S_o} + \omega_{S_o} + \Omega_{S_o}$$

yields the desired results, after substituting Equations 2.22, 2.23, and 2.25, and using the relation

$$L_o = M_o + \omega_o + \Omega_o .$$

c. Long-period Perturbations

Long-period perturbations are included as follows:

(1) Vertical component of the eccentricity vector with respect to the line of nodes vector (a_{yNSL}). This is expressed by

$$a_{yNSL} = e \sin \omega_{S_o} - \frac{1}{2} \frac{J_3 a_E}{J_2 p} \sin i_o = e_L \sin \omega \tag{2.26}$$

where e_L and ω represent the resulting eccentricity and argument of perigee (see Figure 2.3). Equation 2.26 is derived from Brouwer (1959) for long period perturbations.

(2) The modified mean "mean" longitude (L). This is expressed by

$$L = L_S - \frac{1}{4} \frac{J_3 a_E}{J_2 p} a_{xNSL} \sin i_o \left[\frac{3 + 5 \cos i_o}{1 + \cos i_o} \right] \tag{2.27}$$

where

$$a_{xNSL} = e_L \cos \omega \tag{2.28}$$

represents the horizontal component of the eccentricity vector with respect to the line of nodes vector (see Figure 2.3). Equations 2.27 and 2.28 are derived from Brouwer (1959) with Lyddane (1963) modifications for low eccentricities or inclinations.

(3) Kepler's equation. Kepler's equation is solved iteratively for the sum of the eccentric anomaly and the resulting argument of perigee $(E + \omega)$ where

$$(E + \omega)_{i+1} = (E + \omega)_i + \Delta(E + \omega)_i \tag{2.29}$$

with

$$\Delta(E + \omega)_i = \frac{U - a_{yNSL} \cos(E + \omega)_i + a_{xNSL} \sin(E + \omega)_i - (E + \omega)_i}{-a_{yNSL} \sin(E + \omega)_i - a_{xNSL} \cos(E + \omega)_i + 1}, \tag{2.30}$$

$$U = L - \Omega_{S_o}$$

and

$$(E + \omega)_1 = U.$$

Equation 2.30 is derived from the relation (Roy, 1988, p.85)

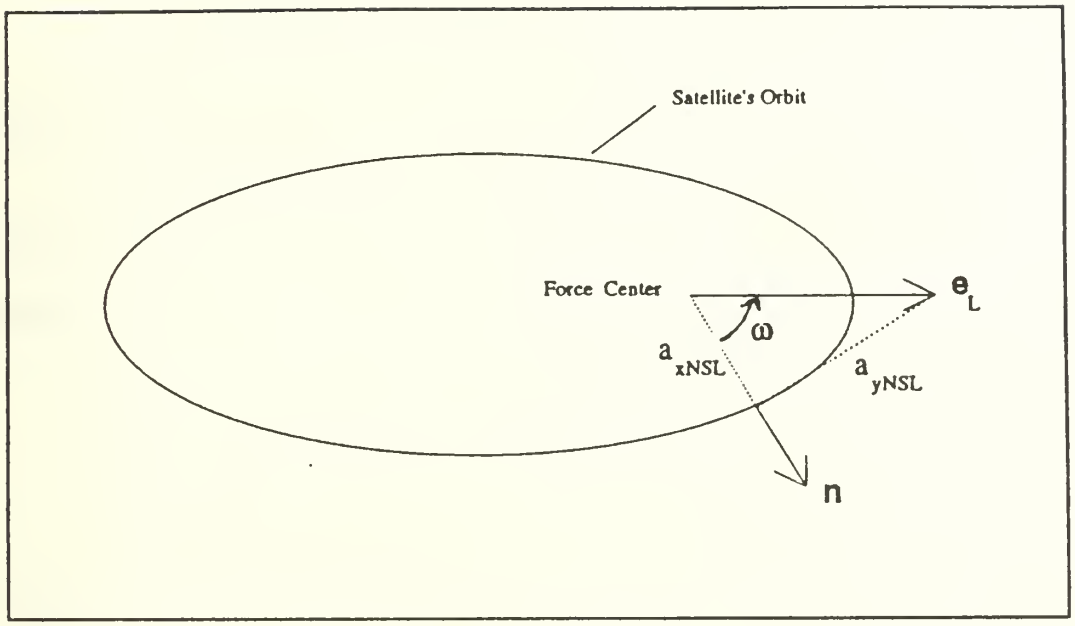

Figure 2.3 Eccentricity Vector and Node Vector

$$\Delta(E+\omega)_i = \frac{L - \Omega_{S_e} + e_L \sin E_i - (E+\omega)_i}{1 - e_L \cos E_i}$$

since

$$e_L \sin E_i = a_{xNSL} \sin(E+\omega)_i - a_{yNSL} \cos(E+\omega)_i$$

and (2.31)

$$e_L \cos E_i = a_{yNSL} \sin(E+\omega)_i + a_{xNSL} \cos(E+\omega)_i .$$

(4) Modified eccentricity. The square of the eccentricity can be expressed by

$$e_L^{\,2} = (a_{xNSL})^2 + (a_{yNSL})^2 . \tag{2.32}$$

This equation simply comes from Equation 2.31.

(5) Modified semilatus rectum. This is expressed by

$$p_L = a\left(1-e_L{}^2\right). \tag{2.33}$$

(6) Radial distance from Earth to satellite. This is expressed by

$$r = a\left(1-e_L\cos E\right). \tag{2.34}$$

(7) Time rate of change of the radial distance. This is expressed by

$$\dot{r} = k_E\frac{\sqrt{a}}{r}e_L\sin E. \tag{2.35}$$

This equation is derived by differentiating Equation 2.34 giving

$$\frac{dr}{dt} = ae_L\sin E\frac{dE}{dt}$$

where (Danby, 1962, p.131)

$$\frac{dE}{dt} = \frac{k_E}{\sqrt{a^3}}\frac{1}{1-e_L\cos E} = \frac{k_E}{\sqrt{a^3}}\frac{a}{r} = \frac{k_E}{r\sqrt{a}}.$$

(8) Product of radial distance and time rate of change of the speed of the satellite, v. This is expressed by

$$r\dot{v} = k_E\frac{\sqrt{p_L}}{r}. \tag{2.36}$$

This equation is true since (Danby, 1962, p.130)

$$\frac{dv}{dt} = \left(1+e\cos v\right)^2\frac{k_E}{\sqrt{p_L{}^3}} = \frac{p_L{}^2}{r^2}\frac{k_E}{\sqrt{p_L{}^3}} = \frac{k_E\sqrt{p_L}}{r^2}. \tag{2.37}$$

(9) Argument of perigee (ω). This is expressed implicitly by

$$\sin u = \frac{a}{r}\left[\sin\left(E+\omega\right) - a_{yNSL} - a_{xNSL}\frac{e_L\sin E}{1+\sqrt{1+e_L{}^2}}\right] \tag{2.38}$$

and

$$\cos u = \frac{a}{r}\left[\cos(E+\omega) - a_{xNSL} + a_{yNSL}\frac{e_L \sin E}{1+\sqrt{1-e_L^{\,2}}}\right], \tag{2.39}$$

where

$$u = \upsilon + \omega.$$

Equation 2.38 is derived algebraically using the relations

$$\sin u = \sin\upsilon\cos\omega + \cos\upsilon\sin\omega$$

$$\sin\upsilon = \frac{a}{r}\sqrt{1-e^2}\sin E$$

$$\cos\upsilon = \frac{e_L - \cos E}{e_L \cos E - 1}$$

combined with Equation 2.31. Equation 2.39 is derived similarly. Now, u can be calculated from

$$u = \tan^{-1}\left(\frac{\sin u}{\cos u}\right). \tag{2.40}$$

d. Short-period Perturbations

The short-period perturbations are now included by

(1) $$r_k = r + \frac{1}{4}J_2\frac{a_E^{\,2}}{p_L}\sin^2 i_o \cos 2u \tag{2.41}$$

(2) $$u_k = u - \frac{1}{8}J_2\frac{a_E^{\,2}}{p_L^{\,2}}\left(7\cos^2 i_o - 1\right)\sin 2u \tag{2.42}$$

(3) $$\Omega_k = \Omega_{S_o} + \frac{3}{4}J_2\frac{a_E^{\,2}}{p_L^{\,2}}\cos i_o \sin 2u \tag{2.43}$$

(4) $$i_k = i_o + \frac{3}{4}J_2\frac{a_E^{\,2}}{p_L^{\,2}}\sin i_o \cos i_o \cos 2u. \tag{2.44}$$

These short-period perturbations are derived from Brouwer (1959). They are the result of taking Brouwer's equations for short-period perturbations and dropping

terms with a coefficient of $k_2 e$ or smaller (Hoots, 1975, p . 9).

e. Resulting Position and Velocity Vectors

The unit orientation vectors are calculated by (see Figure 2.4)

$$\begin{aligned} \mathbf{U} &= \mathbf{M}\sin u_k + \mathbf{N}\cos u_k \\ \mathbf{V} &= \mathbf{M}\cos u_k - \mathbf{N}\sin u_k \end{aligned} \tag{2.45}$$

where

$$\mathbf{M} = \begin{Bmatrix} M_X = -\sin\Omega_k \cos i_k \\ M_Y = \cos\Omega_k \cos i_k \\ M_Z = \sin i_k \end{Bmatrix} \tag{2.46}$$

$$\mathbf{N} = \begin{Bmatrix} N_X = \cos\Omega_k \\ N_y = \sin\Omega_k \\ N_Z = 0 \end{Bmatrix}.$$

The position and velocity vectors are then given by

$$\mathbf{r} = r_k \mathbf{U} \tag{2.47}$$

and

$$\dot{\mathbf{r}} = \dot{r}\mathbf{U} + (r\dot{v})\mathbf{V}. \tag{2.48}$$

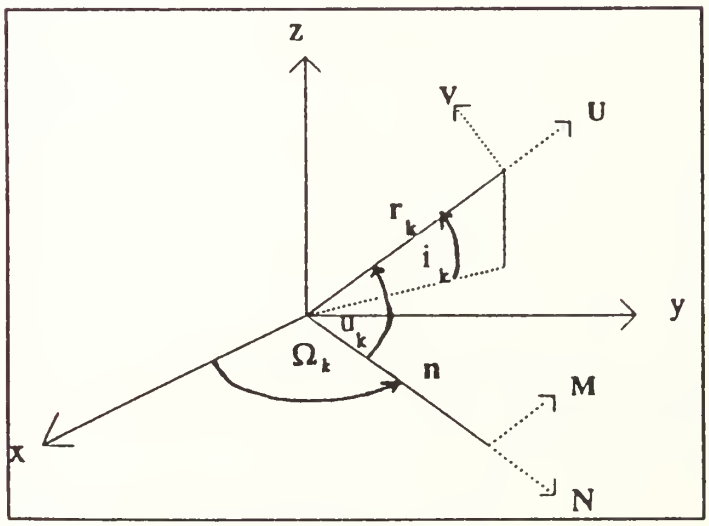

Figure 2.4 Unit Orientation Vectors

III. SGP4 AND SDP4 SATELLITE MOTION MODELS

A. OVERVIEW

The SGP4 and the SDP4 models are the satellite motion models currently used by the AFSPACECOM. Given the six mean orbital elements, an epoch reference time, and a drag factor, SGP4 or SDP4 can predict the position and velocity at a future time. If the satellite is "near-earth" (i.e., orbit period less than 225 minutes), SGP4 propagates the ephemerides, otherwise the "deep-space" satellites are propagated by SDP4. Like SGP, SGP4 considers perturbing accelerations due to the oblateness of the Earth, asymmetry of the Earth's mass about the equatorial plane, and atmospheric drag. The differences between SGP and SGP4 are that SGP4 includes zonal harmonics through J_4 whereas SGP only includes through J_3, SGP4 includes some "deep-space" terms not found in SGP, and SGP4 includes a drag force in its equations of motion where SGP performs a simple correction technique to include the effects of atmospheric drag. The SDP4 model includes the same perturbing effects as SGP4, but also includes effects due to the gravitational attraction of the sun and moon, and some sectoral and tesseral harmonics derived from the longitudinal variations of the Earth's gravitational potential.

In Chapter II classification of satellite motion models was discussed. Like SGP, the SGP4 and SDP4 models solve the equations of motion analytically, and the variation in the satellite's motion is in terms of the changes in the osculating elements with respect to time, thus SGP4 and SDP4 are *general perturbation* models that use *variation of elements.*

B. THEORY

1. SGP4 Model

Here a general discussion is given as presented by Paul Schumacher (1991). SGP4 is based on the theory of Lane and Cranford (1969) which uses the work of Brouwer (1959) and Brouwer and Hori (1961). Brouwer and Hori generalized the original drag-free Brouwer model by including a drag force in the equations of motion. This involved modeling the atmosphere as a spherically symmetric, and non-rotating entity. The density is assumed to decrease exponentially with altitude, providing two free parameters (reference density and scale height) which can be chosen to represent the real atmosphere over some altitude range. This atmospheric model is combined with a gravitational force model which includes zonal harmonics J_2, J_3, and J_4. The equations of motion are written in Delaunay variables and treated by von Zeipel's method. Unfortunately, this model was discovered to be very inaccurate for satellites with low perigee altitudes, exactly where the satellite is nearing decay.

The work of Lane (1965) and Lane and Cranford (1969) attempted to overcome the problems of inaccuracy presented by the Brouwer and Hori model. As a possible improvement to the exponential density model, Lane assumed that the scale height parameter varies linearly with altitude. This assumption led to a new description of the density, namely, an "almost-power" function (Fitzpatrick, 1970). This function involves three parameters: reference density, reference scale height, and scale height gradient. The gravitational model includes zonal harmonics J_2, J_3, J_4, and J_5. Included in the solution are complete first-order short-periodic and secular terms, some second-order secular terms, and all first-order long-periodic terms, except for the mean anomaly which only includes drag-related terms. Singularities resulting from an eccentricity or inclination equal to zero are avoided by Lyddane's approach (1963) of replacing Delaunay variables with Poincare variables. In 1971 Cranford established these results

as the SGP4 theory, and by 1976 SGP4 had replaced SGP as the operational theory at the AFSPACECOM. Like SGP, SGP4 uses similar compromises in its implementation. All J_5 terms, all $O(10^{-3})$ short-period terms and several long-period terms are not included in the solution.

The satellite drag density is measured by the modified ballistic coefficient, B^*. This coefficient is a product of satellite drag coefficient, area-to-mass ratio and a combination of density profile parameters. The modified ballistic coefficient, B^*, is obtained from a differential correction process and is one of the input parameters to SGP4. It is somewhat analogous to the $\frac{dn}{dt}$ parameter used for SGP. Due to the implementation compromises and the difficulty in acquiring accurate density values from a simple model, SGP4 is considered inaccurate for rapidly decaying satellites.

2. SDP4 Model

Here, again, a general discussion is given as presented by Paul Schumacher (1991). SDP4 includes most of the SGP4 implementation, but also includes some deep-space perturbations developed by Hujsak (1979). As mentioned previously, SDP4 only applies to satellites with period greater than 225 minutes. This model includes leading terms for lunar and solar attraction, as well as several longitudinal dependent Earth geopotential harmonics. Secular lunar and solar perturbations are obtained by averaging over the satellite period, and the remaining short-period terms are neglected. Singularities are found in the secular formulae at zero values of inclination and are dealt with by omitting sensitive terms when the inclination is relatively small. A second averaging over the periods of the moon and sun respectively provides additional secular terms and some short-period terms. Earth geopotential resonance corrections are used only for in-track position. Although SDP4 has its shortcomings in its calculations of the perturbations, it is the only "deep-space" analytical model that is used operationally.

IV. PARALLEL COMPUTING

This chapter closely follows the parallel computing chapter written by Warren E. Phipps in his thesis on the parallelization of the NAVSPASUR model, PPT2 (1992).

A. OVERVIEW

1. Definition

As scientific models become increasingly more complex and detailed, the computer requirements in terms of amount of computation and speed become more demanding. Computer engineers have responded by taking two approaches to achieve faster performance.

The first approach is to increase the speed of the circuitry. Although great advances have been made in this area, the speed is bounded by the speed of light. Also, the design and manufacture requirements for continued increases in speed are very costly. The second approach is the use of parallel computing. *Parallel computing* or *parallel processing* provides an alternate means to achieve faster computer performance at a more affordable price. In this new field, various definitions in how to define parallel computing exist. One definition which describes parallel computing most completely and concisely may be taken from (Hwang, 1984, p.6):

> *Parallel processing* is an efficient form of information processing which emphasizes the exploitation of concurrent events in the computing process. Concurrency implies parallelism, simultaneity, and pipelining. Parallel events may occur in multiple resources during the same time interval; simultaneous events may occur at the same time instant; and pipelined events may occur in overlapped time spans. These concurrent events are attainable in a computer system at various processing levels.

In his book, Hwang describes the processing levels. The top level is referred to as the program level which involves executing multiple programs by means of multiprogramming, time sharing, and multiprocessing. This depth of parallel processing is beyond the scope of this thesis. The lower levels (i.e., task level, inter-instruction level, and intra-instruction level) involve the execution of a single program which are applicable to the programming done for this research. Thus, for the purpose of this thesis, *parallel computing* is defined as the efficient form of information processing emphasizing the concurrent computations and manipulation of data to solve a single problem.

2. Classification of Parallel Computers

a. Type Classifications

Implicit in the definition of parallel computing are three methods to achieve parallelism: temporal parallelism, spatial parallelism, and asynchronous parallelism (Hwang, 1984, p. 20). These methods provide a means to classify the various types of parallel computers.

The first type is a pipeline computer. *Pipeline computers* perform overlapped computations , thus use temporal parallelism. Computations are broken into a number of segments or stages where the output of one segment is the input of another segment. If all the segments work at the same speed, the work rate of the pipeline is equal to the sum of the work rates of the segments, once the pipe is full. This is analogous to an assembly line in a factory. An example of a pipeline computer is a Cray-1.

The second type is a processor array. A *processor array* is a set of identical synchronized processing elements to achieve spatial parallelism. It is capable of simultaneously performing the same operation on different data. An example of a processor array is the Connection Machine.

The third type is a multiprocessor. The term *multiprocessor* refers to a shared-memory multiple-CPU computer designed for parallel processing. These CPU's or processors are capable of performing independent operations and may achieve asynchronous parallelism (i.e., the processors have the ability to work with the most recently available data). An example of a multiprocessor is the Cm* of Carnegie-Mellon University.

The final type is a derivative of the multiprocessor, the multicomputer. The *multicomputer* is a multiple CPU computer designed for parallel processing without the shared memory. Multicomputers can also achieve asynchronous parallelism through communications between the processors or nodes. An example of a multicomputer is the INTEL iPSC/2 hypercube. Since each processor has its own memory and can perform independent operations, multicomputers offer a greater degree of freedom in programming. However, communication between the nodes may require that synchronization be programmed into the multicomputer code to achieve the objective of the code.

The four type classifications are not necessarily mutually exclusive. For example many commercially available processor arrays, multiprocessors, and multicomputers use pipeline processors to complete operations such as vector processing.

b. Architectural Classifications

Parallel computers can also be classified according to their architecture. One such classification was devised by Flynn (1966). He categorized digital computers by the multiplicity of hardware used to manipulate instruction and data streams. Four classes of computers resulted from this :

1. **S**ingle-**I**nstruction stream, **S**ingle **D**ata stream (SISD). Most serial computers fall into this category. Although instructions are completed sequentially, this category includes overlapping instructions (pipelining). Therefore, pure pipeline processors also belong to this category.

2. **S**ingle **I**nstruction stream, **M**ultiple **D**ata stream (SIMD). Processor arrays fall into this category. The processor array receives a single set of instructions, but each element receives and manipulates its own set of data.

3. **M**ultiple **I**nstruction stream, **S**ingle **D**ata stream (MISD). No current computers fall into this category.

4. **M**ultiple **I**nstruction stream, **M**ultiple **D**ata stream (MIMD). Most multiprocessors and multicomputers fall into this category. The INTEL iPSC/2 is a MIMD machine.

c. Topological Classifications

Another classifying scheme for parallel computers that only apply to processor arrays, multiprocessors, and multicomputers concerns the topology of the inter-processor connections. These connections are the means through which processors can communicate with one another. Seven important processor organizations are the mesh, the pyramid, the fat tree, the butterfly, the shuffle-exchange, cube-connected cycles and the hypercube. Figure 4.1 show examples of the first six processor organizations mentioned. Figure 4.2 shows examples of the hypercube topology. This thesis will obviously emphasize the hypercube topology. For a more complete discussion on the other topologies see (Quinn, 1987, pp. 25-30).

3. Measurements of Performance

The objective of parallel computing is to provide faster computation, thus certain defined parameters are needed to measure the performance of the parallel computer versus the serial computer. Computation speeds depend on many factors such as hardware design, technical specifications of the computer components, and the design of the algorithm to execute the computations. Two common measures of effectiveness that account for both the hardware and the algorithm design are *speedup* and *efficiency*.

Speedup, S_p, is defined as the ratio between the time needed to execute the most efficient sequential algorithm to perform a set of computations, T_S, and the time to perform these same computations using parallelism, T_p,

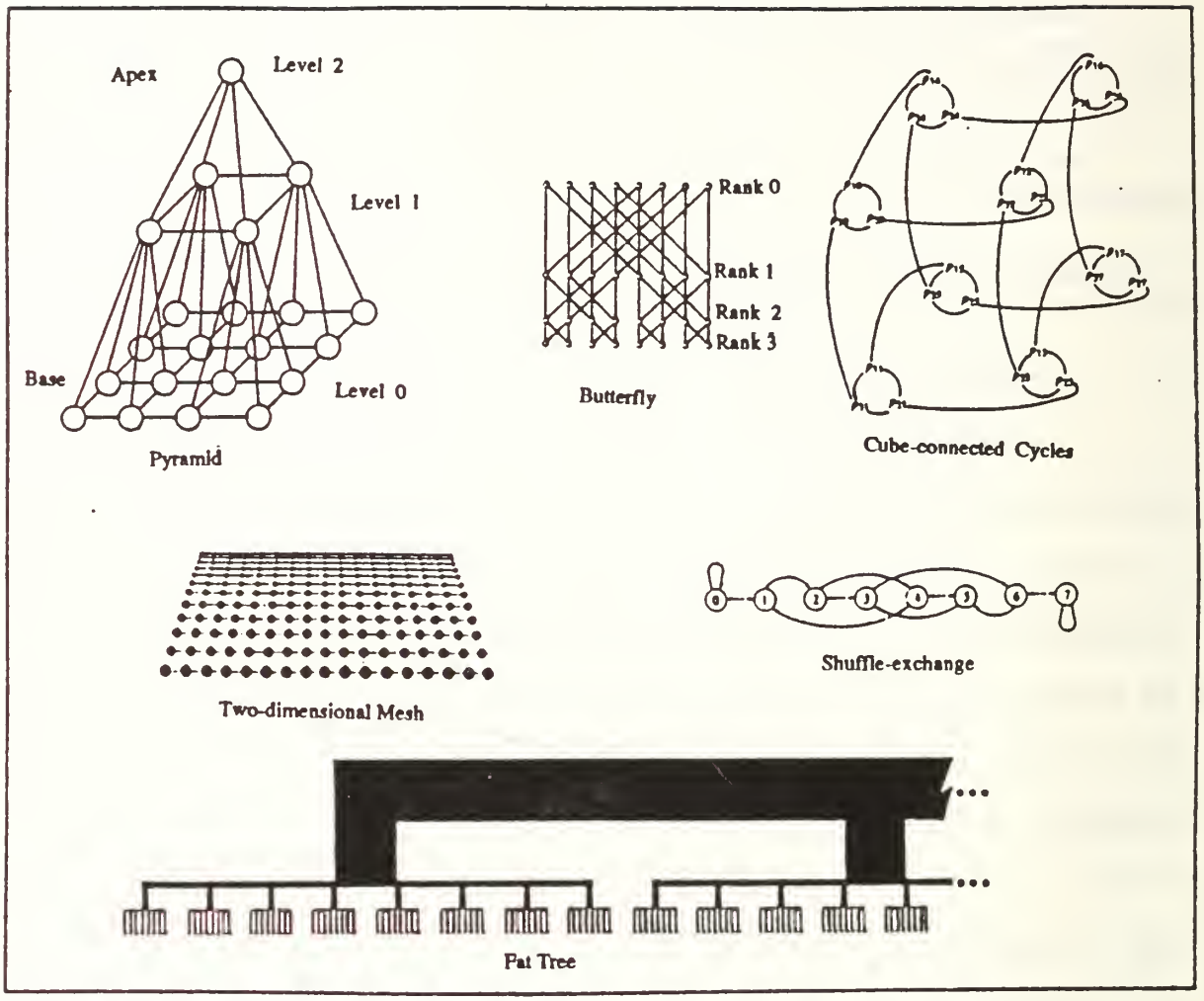

Figure 4.1 Various processor organizations

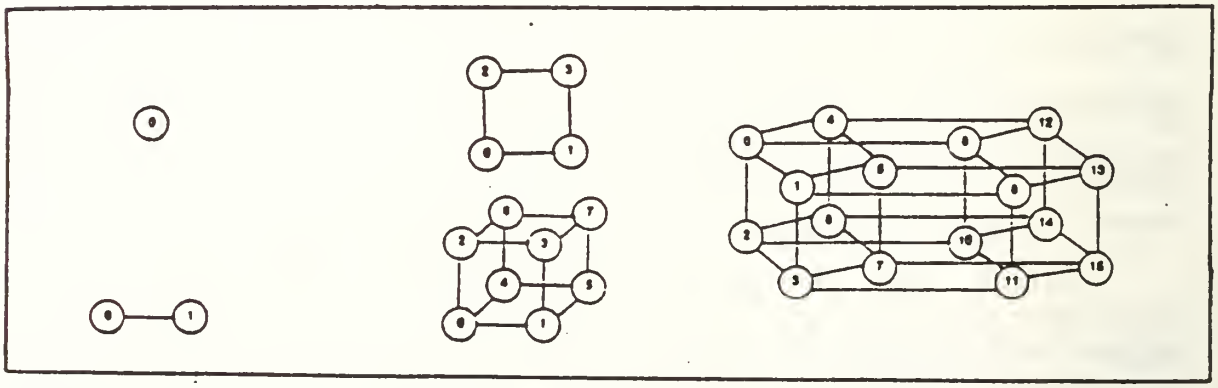

Figure 4.2 Hypercubes of dimension zero through four

$$S_p = \frac{T_S}{T_p}. \tag{4.1}$$

It should be emphasized that even though the serial program and the parallel program execute the same set of computations, the two programs do not necessarily follow the same algorithm. Parallel programs often include additional instructions or operations to accommodate parallelism. Also, as previously mentioned, the serial program should be the most efficient obtainable, so as not to be misleading in the calculation of *speedup*. Many suggest that the times, T_p and T_S, be measured using a specific parallel computer and the fastest serial computer. But, the variation in the technical specifications between the two computers make the comparison unclear, thus do not provide an effective means of assessing the effectiveness of parallel computing. Therefore, for the purpose of this thesis, S_p is defined as the ratio of time, T_1, taken by the parallel computer to execute the most efficient serial algorithm and the time, T_p, taken by the same parallel computer executing the parallel algorithm using p processors,

$$S_p = \frac{T_1}{T_p}. \tag{4.2}$$

The other measure, *efficiency*, E_p, is defined as the ratio of the speedup to the number of processors,

$$E_p = \frac{S_p}{p}. \tag{4.3}$$

Efficiency accounts for the relative cost in terms of number of processors required in achieving a certain speedup.

Many factors can limit the possible speedup and efficiency of a parallel program. These factors include the number of sequential operations that cannot be parallelized, the communication time among processors, and the time each processor is idle due to synchronization requirements. Many have argued that these factors severely limit the

benefits of parallel computing. Regardless of these factors, research demonstrates parallel computing as an effective means to reduce computation time (Quinn, 1987, pp. 18-20). If one considers only the number of sequential operations in a program that cannot be parallelized, Amdahl's Law indicates that the maximum speedup achievable by p processors is,

$$S_p \leq \frac{1}{f + (1-f)/p}, \tag{4.4}$$

where f is the fraction of operations that must be performed sequentially (Amdahl, 1967, pp. 483-485). Equation 4.4 provides an initial means to see if a given algorithm is a good candidate for parallelization.

B. INTEL iPSC/2 HYPERCUBE

The design of a parallel algorithm is done with a specific parallel computer in mind. This design involves maximizing the speedup and efficiency. In determining the parallel computing potential of the AFSPACECOM satellite motion models, SGP and SGP4, an INTEL iPSC/2 hypercube computer, located at the Department of Mathematics at the Naval Postgraduate School, was used. This computer is a MIMD multicomputer with a hypercube topology. It consists of a system resource manager called the host, and eight individual processors, referred to as nodes. The host is a 386-based computer, which can process data as well as providing an interface for the user.

The nodes are complete and independent INTEL 80386 microprocessors. Each node also contains a 80387 numeric coprocessor, its own local memory, and a Direct-Connect Communications Module (DCM). Each node may also contain a Vector Extension (VX) module for pipelined vector operations. The iPSC/2 located at the Naval Postgraduate School has only one node that contains the VX module.

Communications among the nodes and host are done with message passing. The DCM permits messages to be sent from an individual node to a receiving node directly without disturbing the remaining nodes. Each node along the message path can be thought of as a switch which simply closes when a message is sent. Other hypercube designs require messages to be stored and forwarded with each node along a message path until the message is received by the destined node.

The iPSC/2 uses a UNIX operating system and the available programming languages are Fortran or C. For a more detailed listing of the technical specifications of the INTEL iPSC/2 hypercubes see Appendix A.

C. METHODS OF PARALLELIZATION

1. Vectorization

Vectorization is the process of converting blocks of sequential operations into vector instructions that may be pipelined. A simple example of vectorization using Fortran is the following:

Sequential Code:

$$\begin{aligned} & Do\ 10 \quad i = 1, N \\ 10 \quad & z(i) = x(i) + y(i) \end{aligned}$$

Vector Code (VAST2):

$$call\ vadd(N, x, 1, y, 1, z, 1)$$

VAST2 is an example of a vectorization compiler to assist in the vectorization of a serial program. It is used by the iPSC/2 at the Naval Postgraduate School. There are many other vectorization compilers that are also available commercially. (Quinn, 1987, pp. 233-235)

Vectorizing compilers automatically vectorize serial codes before execution. Some vectorizing compilers may even indicate to the user areas of the program that limit

potential vectorization. Unfortunately, vectorizing compilers are not solely capable of maximizing vectorization. Most vectorizing compilers are restricted in their ability to identify sequential blocks that can be vectorized and the resulting vectorization may not always be straight forward. The VAST2 compiler, for example, supports only Fortran programs and is only able to vectorize *do loops* and *if statements*. (iPSC/2 VAST2 User's Guide, 1989)

2. Distributing Computations

Vectorization illustrates the first level of parallelism, in that tasks are parallelized on individual processors. On the other hand, to distribute partitions of a program to different processors on a multicomputer another approach is needed. Compilers created to achieve this higher level of parallelism have not been successfully implemented as with the vectorization compilers. Thus, the task of efficiently parallelizing an algorithm is left to the user.

Since the performance of parallel algorithms depend on factors such as processor speed, memory access time, and memory capacity, in other words, the technical specifications of the parallel computer, the process of parallelizing an algorithm must be done with a particular parallel computer in mind. The multicomputer where each node has its own memory presents the greatest flexibility to the user. Here, the user has to partition the problem among the processor nodes. The hypercube topology allows the user to use the natural topology of a given problem to divide the problem into parallel *processes*. A *process* is defined as a single statement or a group of statements which are a self-contained portion of the total computations. For the INTEL iPSC/2, two decomposition strategies are suggested: *Control Decomposition* and *Domain Decomposition*. (iPSC/2 User's Guide, 1990, pp. 4-1 - 4-6)

a. Control Decomposition

Control Decomposition is the method of dividing tasks or processes among the individual processors or nodes, a divide and conquer approach to the problem. This method is recommended for problems with irregular data structures or unpredictable control flows.

One way control decomposition is implemented is by using a self-schedule approach. One node acts as the manager while the remaining nodes act as workers. The managing node maintains a list of processes to be accomplished by the working nodes and distributes these processes accordingly. The working nodes request jobs, receive processes, and perform the given task. Of course, in the self-scheduling method the cost is one processor, namely, the managing processor. (iPSC/2 User's Guide, 1990, p. 4-4)

A second method of control decomposition involves the pre-scheduling of the processes. The exact tasks of each processor are explicitly stated in the parallel program. Although, this method saves the cost of one node, a great deal of care must be taken to ensure the processes are distributed as evenly as possible among the nodes.

b. Domain Decomposition

Domain Decomposition involves dividing the input data or domain among the nodes. These partitioned sets may be specific data sets such as blocks of a matrix or may represent a specific grid such as used in finite difference or finite element methods to solve partial differential equations. Unlike control decomposition, domain decomposition requires that each node perform essentially the same tasks but with different input data.

Domain decomposition is suggested if a program contains calculations based on a large data structure and if the amount of work is the same for each node. For example, the multiplication of two large matrices by block multiplication uses domain decomposition. Although domain decomposition may appear to be perfectly

parallelizable, the user must use caution to ensure each input set requires about the same amount of work.

3. Improving Performance

The parallelization of a problem may require the use of control decomposition, domain decomposition, or a combination of both methods to be an efficient algorithm. Once a particular method is chosen, several factors need be considered to improve the performance of the parallel algorithm. These factors include *load balance*, *communication to computation ratio*, and *sequential bottlenecks*.

a. Load Balance

Load balance refers to the degree to which all nodes are working or active. In the case where the work is not evenly distributed among the nodes, the parallel algorithm will be limited in its speedup ability. Ways to achieve load balancing is through a decrease in grain size of the parallel tasks, self-scheduling tasks, or redistributing the domain. *Grain size* refers to the relative amount of work completed in parallel. For example, distributing computations may be considered as large grain parallel computing while pipelined vector operations are small grain parallel computing.

b. Communication to Computation Ratio

Communication to *computation ratio* is the ratio of the time spent communicating to the time spent computing. The time loss for communications is unavoidable in parallel algorithms, except for perfectly parallel problems. A large communication to computation ratio restricts the performance of a parallel program. The objective is to maximize the time a node spends computing, thus minimizing the time it is communicating. Reducing the communication to computation ratio may be accomplished by increasing the grain size, grouping messages, or recalculating values vice receiving the value from another node.

c. Sequential Bottlenecks

In some cases a task cannot begin unless a previous task is completed, thus limiting the number of tasks that can be done in parallel. A *sequential bottleneck* is the situation where processors are waiting for another processor to complete a task before they may continue. The fraction of operations of a algorithm that cannot be done in parallel can greatly limit speedup as seen by Amdahl's Law (Equation 4.4). Sequential bottlenecks are potentially found with any requirements of the nodes to synchronize. The only way to remove sequential bottlenecks is to modify or restructure the algorithm in order to overlap sequential code with other computations.

V. PARALLELIZATION OF SGP, SGP4, AND SDP4

The objective of this thesis is to assess the parallel computing potential of the AFSPACECOM's satellite motion models SGP, SGP4, and SDP4. This analysis may be done by comparing the speedups and efficiencies of different parallel algorithms designed using the various methods and strategies discussed in Chapter IV. Warren Phipps (1992) concluded in his thesis that vectorization and control decomposition were not beneficial in reducing the computation time of the NAVSPASUR model (PPT2), whereas the domain decomposition strategy showed promise. Since like the PPT2 model, the SGP, SGP4, and SDP4 models are analytic and have computation times of the same order of magnitude; it was decided that this thesis would only investigate the domain decomposition strategy for all three models.

The objective of the domain decomposition method is to the reduce the SGP, SGP4, and SDP4 models' computation time by the concurrent computation of several satellite data sets. Each node of the hypercube performs identical tasks on different satellite data sets, at the same time. Thus, as was considered with PPT2, the ultimate goal is to reduce the overall computation time for several objects in orbit.

A. ALGORITHM

The parallel algorithm design used for SGP, SGP4, and SDP4 is identical to the parallel design used by Phipps (1992) to achieve the domain decomposition of PPT2. A single node is devoted to reading all the input data and distributing this data to the working nodes (i.e., those processors that actually propagate the given input data by either calling the SGP, SGP4, or SDP4 subroutine to produce a position and velocity vector for each time required beyond epoch), and a single node is to collect the results

and write them to an output file. Figure 5.1 illustrates how the satellite data is distributed for a three dimensional hypercube.

In his thesis, Phipps presented the advantages and disadvantages of using this particular parallel algorithm design:

advantages:

* Relatively simple to apply. No requirement for communication or synchronization among the working nodes.

* The existing subroutine (i.e., PPT2 for Phipp's thesis, and SGP, SGP4, or SDP4 for this thesis) may be used with only minor modifications. The same input parameters can be used for the parallel program as for the serial program.

* The strategy to achieve maximum efficiency is reduced to developing an algorithm to distribute the data in a timely manner. Thus, decreasing the wait time for any one node.

disadvantages:

* Potential for sequential bottlenecks at input/output portions of algorithm. Reading and writing to an external file is extremely time consuming. With the iPSC/2 hypercube used for this thesis and Phipp's thesis the input/output is completed sequentially.

* The cost is the loss of two nodes, the input node and the output node.

A complete listing of the source codes, and input/output parameters for the parallelized versions of SGP, SGP4, and SDP4 are contained in Appendix B.

The serial algorithms used for this research for SGP, SGP4 and SDP4 are minor modifications of the original algorithms obtained from the AFSPACECOM. The modified serial algorithms read the entire batch of satellite data before propagating any of the input. For example, suppose 1000 satellites need to be propagated, the modified serial algorithm would read the input data for all 1000 satellites and proceed to propagate the position and velocity vectors one right after the other for each satellite. On the other

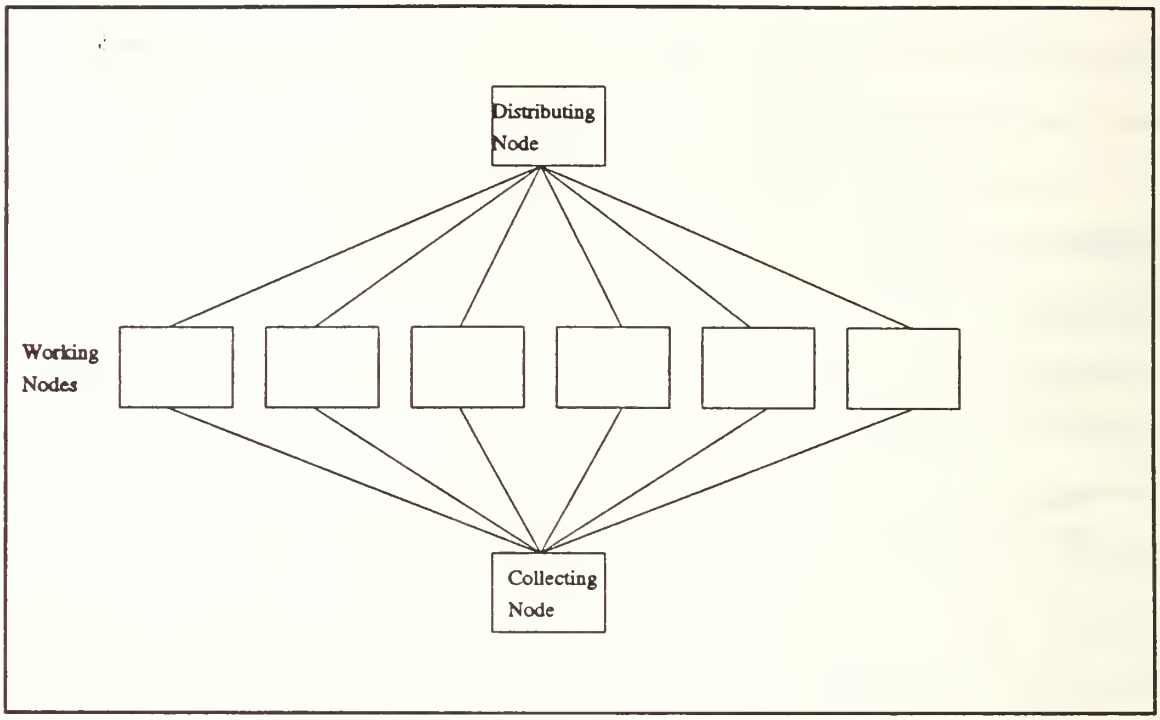

Figure 5.1 Distribution of Satellite Data

hand, the original serial algorithms read in one satellite, propagate the data, read in the next satellite, and propagate its data, etc. This modification was performed to create more of a similarity between the serial algorithms and the parallel algorithms. It should be noted that the execution times of the modified serial algorithms are equivalent to the original serial algorithms.

The execution times for the serial and parallel programs for each of the models (SGP, SGP4, and SDP4) were obtained by using the internal clocks contained in each processor. For the parallel programs, the processor that depicted the largest amount of time for execution was used. Each execution time is the result of an average of ten recorded execution times.

1. Assessment of SGP

a. Results

The experimental results in the analysis of the parallelization of SGP depict similar trends and conclusions as obtained by Phipps (1992) in his analysis of PPT2. Figure 5.2 is a plot of the mean execution time for the serial version and the parallelized versions of SGP, using a four-node and an eight-node hypercube, versus the number of satellites propagated. See Appendix C for the execution times. As seen by Figure 5.2, the parallelized version of SGP was successful in reducing overall computation time. For convenience the serial version of SGP will be referred to as SSGP and the parallel version as PSGP. This notation will also be used in the assessment of SGP4 and SDP4. Table 5.1 gives the efficiencies and speedups for different numbers of satellites using four nodes and eight nodes. This table depicts a significant increase in efficiency using eight versus four nodes. This increase in efficiency presents the possibility of achieving even higher efficiencies with the use of PSGP for higher dimensional hypercubes. Table 5.1 also indicates an approximate three to one ratio in the speedups obtained using eight nodes versus four nodes. This is expected since the eight node hypercube contains six "working" nodes, which is three times larger than the two "working" nodes used in the four node hypercube.

Another trend observed in Table 5.1, more notably for the case with four nodes, is the slight increase in performance to a peak value and then slightly decreasing as the number of satellites increases. For this parallel algorithm the computation to communication ratio does not vary with the number of satellites. In agreement with Phipps (1992), the increase in performance must mainly be due to the lessening impact of the algorithm's overhead on total execution time. This overhead includes some small computations performed by each working node to determine the number of satellites it

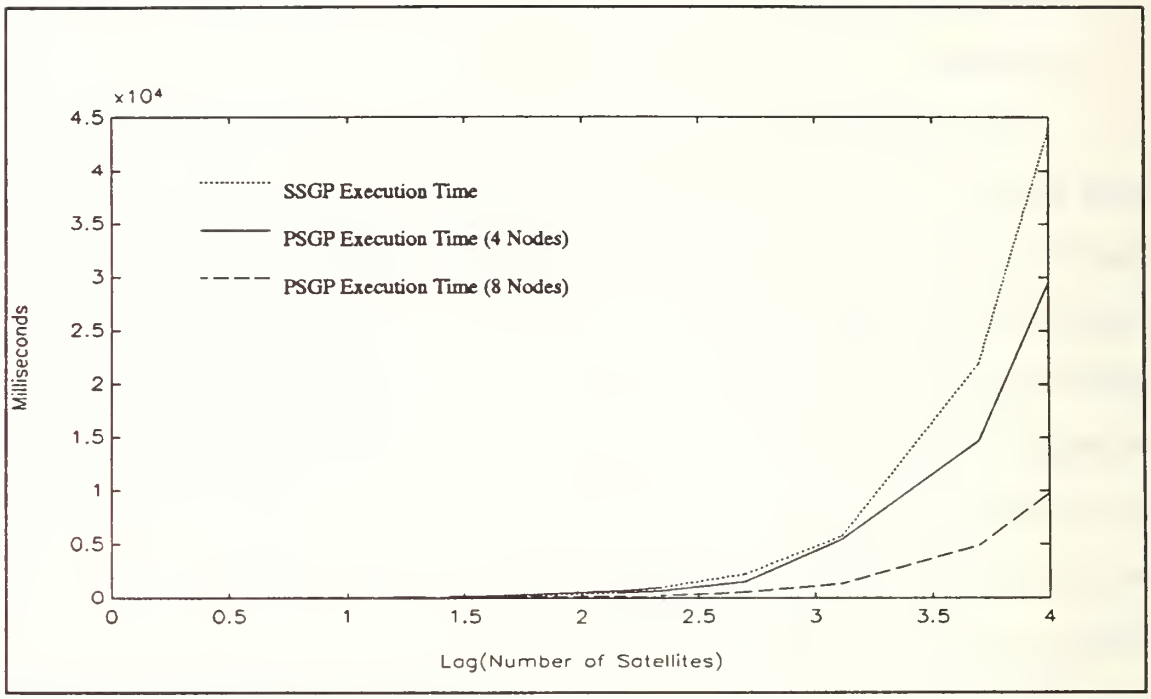

Figure 5.2 SGP Execution Times

TABLE 5.1 PSGP PERFORMANCE

Number of Satellites	Eight Nodes		Four Nodes	
	S_p	E_p	S_p	E_p
6	2.76	.345	1.44	.360
12	3.46	.433	1.47	.368
36	4.11	.514	1.52	.380
144	4.41	.552	1.51	.376
216	4.49	.561	1.51	.377
500	4.49	.561	1.49	.374
1296	4.49	.561	1.41	.352
5000	4.48	.561	1.49	.373
10000	4.47	.559	1.49	.372

must receive from the distributing node. Since these computations are only completed once in the program, the time cost with these calculations become negligible as the number of satellites propagated increases. One possible phenomenon to offset this increase may have to do with the amount of message sets contained in the local buffer of a working node in terms of the time cost to manage these sets (i.e., each message set corresponds to a satellite), which may increase more than linearly with the number of satellites. If a node is not ready to receive a message it stores it in a local buffer. As will be explained in the next section, the use of four and eight nodes result in the working nodes having message sets received from the distributing node stored in their local buffers. For a large number of satellites, several messages will be stored in each local buffer soon after the program is initiated. Although the time for a node to read from its local buffer is insignificant, the time involved in managing several storage sets may not be. The efficiency, in the case of eight nodes, peaks and levels out at .561 for 216 satellites and decreases only slightly to .559 at 10000 satellites. The efficiency for four nodes peaks at .380 for 36 satellites and decreases to .372 at 10000 satellites.

b. Improvements

As stated in the previous section, PSGP may yield increased efficiencies and speedups if applied to a hypercube of greater dimension than three available on the iPSC/2 hypercube used for this thesis. Since the number of working nodes is not fixed for this algorithm, PSGP can be applied to any dimension hypercube without modifications. An increase in efficiency should occur with an increase in the hypercube dimension until the time to distribute a separate satellite data set to each working node exceeds the time required by the node to propagate a single satellite. Hence, the performance of the algorithm could improve by applying it to an optimal dimension hypercube.

Since the hypercube at the Naval Postgraduate School is limited to eight processors, a reasonably accurate model developed by Phipps for his analysis of PPT2 was used to estimate the optimal hypercube dimension. The total execution time for PSGP to propagate n satellites with p processors, $t(p)$, can be modeled by

$$t(p) = t_{w1}(p) + t_{w2}(p) + t_c(p) \tag{5.1}$$

where $t_{w1}(p)$ is the time the last node must wait to receive its first satellite data set, $t_{w2}(p)$ is the total time the last node must wait to receive all of its subsequent satellite data sets, and $t_c(p)$ is the time for each node to propagate its share of the n satellites.

As discussed in the previous chapter, the iPSC/2 uses a Direct-Connect Module (DCM). Because of the DCM the startup time for a message to be passed between two nodes is essentially constant regardless of the length of the path the message is sent through. Thus, the time to send a message between two nodes is only a function of the size or number of bytes contained in the message. Since the size of all the messages between the distributing node and the working nodes are of the same size (104 bytes), the time to send a single message between the distributing node and each working node is essentially constant. In this algorithm there are $p-2$ working nodes. Letting $t_M(1)$ represent the time to send a single message between the distributing node and a working node, $t_{w1}(p)$ may be modeled by

$$t_{w1}(p) = [(p-2)-1]t_M(1) = (p-3)t_M(1). \tag{5.2}$$

To determine $t_M(1)$, a small program was created and executed on the iPSC/2 hypercube at the Naval Postgraduate School. This program timed the message passing between two nodes for various size messages. For each case the time to send a single byte was calculated, and these values were averaged to equal $3.60 \times 10^{-4} \frac{m\sec}{byte}$. This value is equivalent to $2.78 \frac{Mbytes}{\sec}$ which when rounded matches the hardware speed of

2.8 $\frac{Mbytes}{sec}$ quoted in the iPSC/2 technical summary (1988, p.11). Thus, the mean value of $t_M(1)$ is approximately

$$(3.60\times10^{-4})(104) = .0374 \text{ msec.}$$

The total wait time for a working node to receive subsequent satellite data sets, $t_{w2}(p)$, is a function of the elapsed time for the working node to propagate a single satellite data set, the elapsed time for the distributing node to send a subsequent satellite data set to the working node, and the number of satellites the working node must propagate. Since the distributing node distributes the data while the working nodes are computing, the wait time is zero if the subsequent satellite data arrives before the working node is ready to receive it. On the other hand, if the subsequent satellite data arrives after the node finished with the previous satellite data set, the wait time is the difference between the elapsed time for the distributing node to send the node another satellite data set and the computing time for the previous satellite. Since the distributing node must send a satellite data set to each of the other nodes before sending a subsequent data set to the last node, the elapsed time for the distributing node to send another data set can also be modeled by $t_{w1}(p)$ in Equation 5.2. Thus, the total wait time is the wait time for each subsequent satellite data set multiplied by the number of satellite data sets received by each working node. Letting the time to propagate a single satellite be represented by $t1$, $t_{w2}(p)$ may be modeled by

$$t_{w2}(p) = \begin{cases} 0 & \text{if } t_{w1}(p) < t1 \\ \left(\frac{n}{p-2} - 1\right)(t_{w1}(p) - t1) & \text{if } t_{w1}(p) \geq t1. \end{cases} \tag{5.3}$$

The time, $t1$, was measured to be 4.60 milliseconds using SSGP.

Assuming the time for one node to propagate n satellites, $t(1)$, is

$$t(1) = n(t1),$$

the total computation time for each working node, $t_c(p)$, may be approximated by

$$t_c(p) = \frac{n(t1)}{p-2}. \tag{5.4}$$

Substituting Equation 5.1 into Equations 4.2 and 4.3, the speedup and efficiency using p processors can be modeled by

$$\begin{aligned} S_p &= \frac{t(1)}{t(p)} = \frac{n(t1)}{t_{w1}(p)+t_{w2}(p)+t_c(p)} \\ E_p &= \frac{S_p}{p} = \frac{n(t1)}{p\left[t_{w1}(p)+t_{w2}(p)+t_c(p)\right]}. \end{aligned} \tag{5.5}$$

The total execution time, speedup, and efficiency ($t(p)$, S_p, and E_p) were calculated using Equations 5.2 - 5.5 along with the following substitutions:

* $t_M(1)$ = .0374 msecs
* $t1$ = 4.60 msecs
* n = 5000 satellites

The number of satellites was chosen to be 5000 since this seemed to be a reasonable number for SGP's current use at various sensor sites. Figures 5.3 and 5.4 indicate the estimates of $t(p)$, S_p, and E_p using 4 to 1024 processors (a cube dimension of 2 to 10). Using this model, PSGP is capable of achieving a maximum efficiency above .9 using a 7 dimensional hypercube (128 nodes). These plots are only estimates of the true values of speedup and efficiency. Their prediction for the speedup and efficiency using 4 and 8 nodes are slightly higher than the obtained experimental values, but most certainly provide a good indication of the parallel computing potential of PSGP for higher dimensional hypercubes. Notice for 4 and 8 nodes, $t_{w2}(p)$ equals zero resulting in stored messages in the local buffers of the working nodes. Since $t_{w1}(p)$ is much much less than

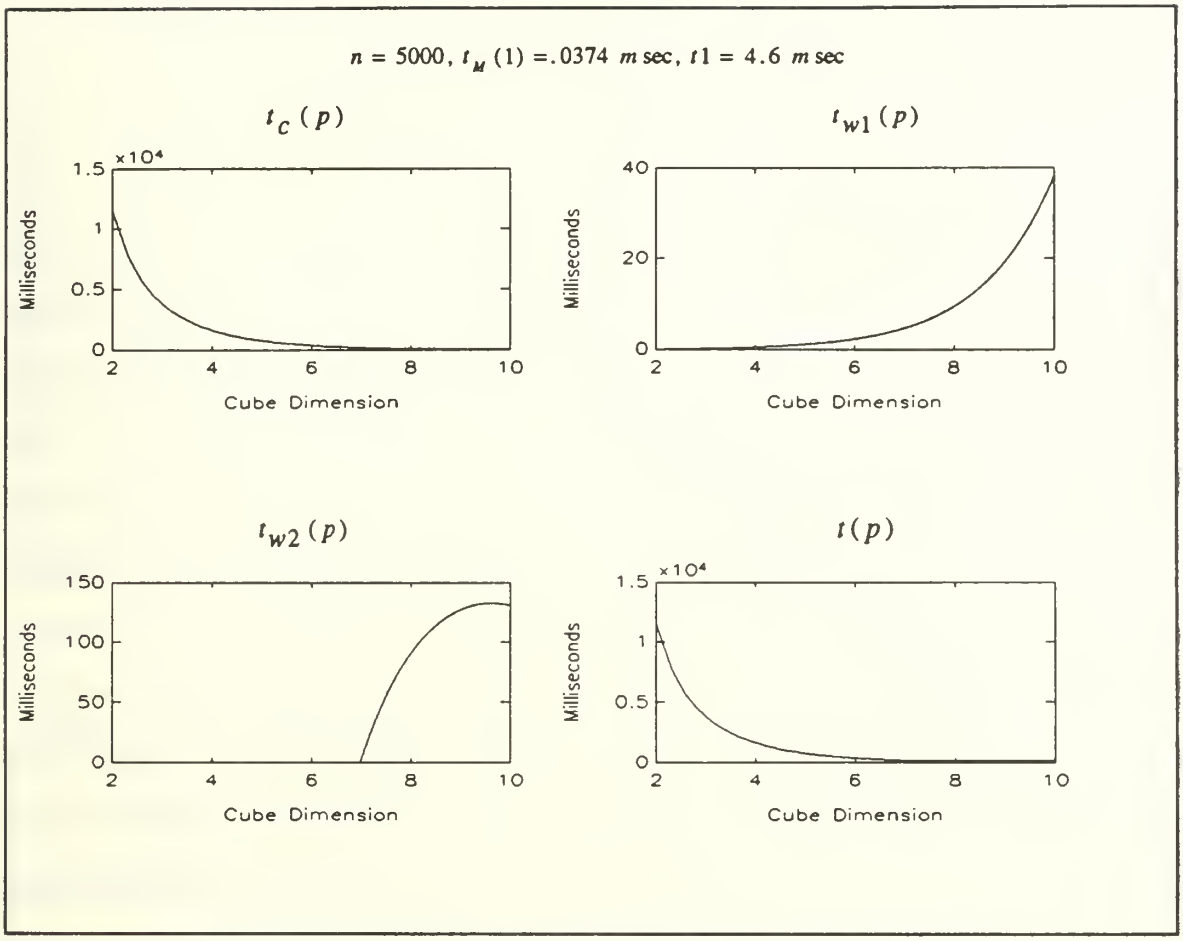

Figure 5.3 Estimated Execution Time of PSGP for Various Hypercube Sizes

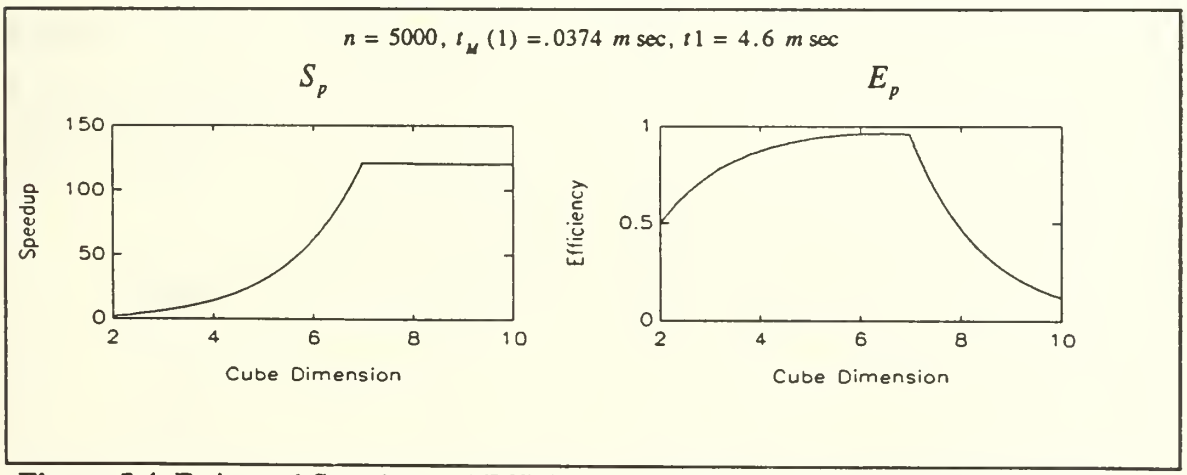

Figure 5.4 Estimated Speedup and Efficiency of PSGP for Various Hypercube Sizes

$t1$, several messages will be stored in the buffers, when propagating a large number of satellites, shortly after the program is initiated.

2. Assessment of SGP4

a. Results

Like SGP, the parallelization of SGP4 also produced similar trends and conclusions as the parallelization of PPT2. Figure 5.5 is a plot of the mean execution time for the serial and the parallelized versions of SGP4, using a four-node and an eight-node hypercube, versus the number of satellites propagated. See Appendix C for the execution times. Table 5.2 gives the efficiencies and speedups for a various number of satellites using four and eight nodes. Again, an approximate three to one ratio exists between the values of speedup using eight nodes versus four nodes. Table 5.2 depicts the same trend as observed with PSGP. In the case of four nodes, the peak value occurs at 36 satellites with an efficiency of .400. For eight nodes this occurs at 500 satellites with an efficiency of .595. Notice again an increase in efficiency as the number of nodes increased from four to eight, motivating the investigation of using hypercubes of higher dimension.

b. Improvements

The model used in the previous section to estimate an optimal dimension hypercube is also used here, since PSGP uses the domain decomposition parallel algorithm. The total execution time, speedup, and efficiency ($t(p)$, S_p, and E_p) were calculated using Equations 5.2-5.5 with the following substitutions:

* $t_M(1) = .0634$ msec (the message size between the distributing node and the working nodes is 176 bytes).
* $t1 = 6.60$ msecs
* $n = 5950$ satellites

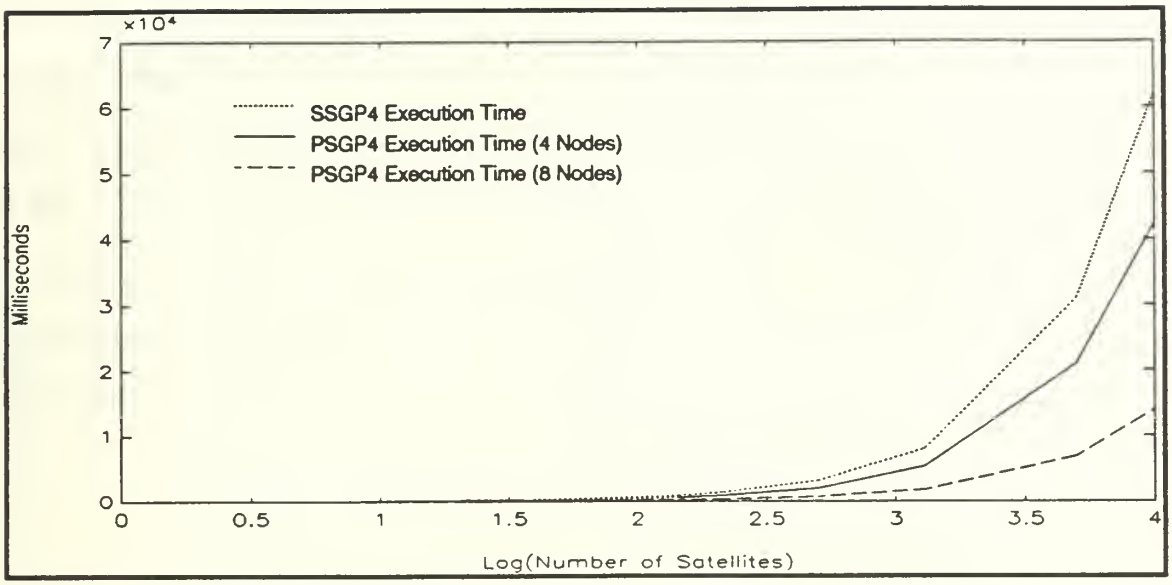

Figure 5.5 SGP4 Execution Times

TABLE 5.2 PSGP4 PERFORMANCE

Number of Satellites	Eight Nodes		Four Nodes	
	S_p	E_p	S_p	E_p
6	3.31	.414	1.55	.388
12	3.85	.481	1.58	.394
36	4.45	.556	1.60	.400
144	4.71	.588	1.60	.399
216	4.74	.592	1.59	.398
500	4.76	.595	1.57	.393
1296	4.73	.592	1.51	.377
5000	4.54	.568	1.48	.369
10000	4.44	.555	1.47	.367

The number of satellites was chosen based on the information received from the AFSPACECOM. The AFSPACECOM in Colorado Springs propagates data on approximately 7000 satellites a day, and about 85 percent of those are low earth orbit, yielding the 5950 value. Figures 5.6 and 5.7 depict the estimates of $t(p)$, S_p, and E_p using 4 to 1024 processors. Using this model, PSGP4 is capable of achieving a maximum efficiency greater than .9 using a 6 dimensional hypercube (64 nodes). As with PSGP, the estimates using this model for PSGP4 predict a slightly better performance than actually achieved for a hypercube of 2 and 3 dimensions.

3. Assessment of SDP4

a. Results

The parallelization of SDP4 was also successful, yielding similar results as the previous parallel algorithms. Figure 5.8 plots the mean execution time for the serial and the parallelized versions of SDP4, using a four-node and an eight-node hypercube, versus the number of satellites propagated. See Appendix C for the execution times. Table 5.3 gives the efficiencies and speedups for different number of satellites using four nodes and eight nodes. As expected the speedups achieved using eight nodes is approximately three times greater than that for four nodes. Again, there are higher efficiencies obtained using eight nodes versus four nodes for any given number of satellites. As expected, peak efficiencies exist as a function of the number of satellites for both the four node and eight node case. In using eight nodes a peak efficiency of .644 is achieved when propagating 500 satellites. In using four nodes a peak efficiency of .433 occurs at 144 satellites.

b. Improvements

Using the same model, $t(p)$, S_p, and E_p were calculated using Equations 5.2 - 5.5 with the following substitutions:

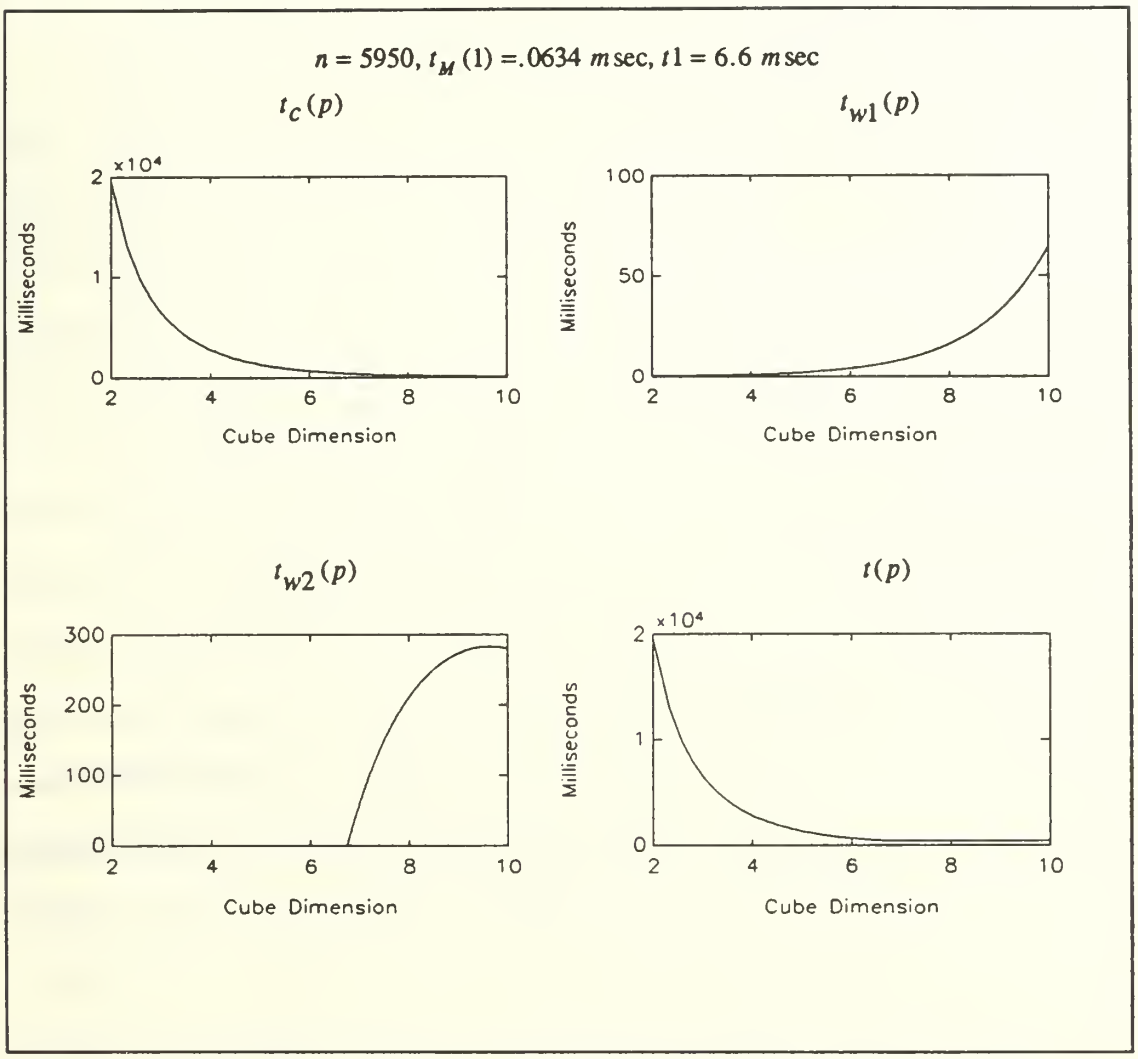

Figure 5.6 Estimated Execution Time of PSGP4 for Various Hypercube Sizes

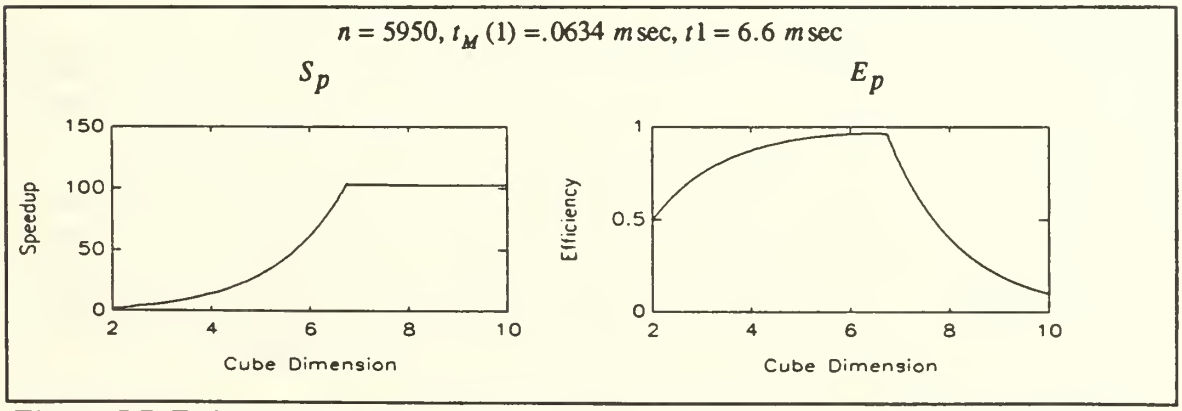

Figure 5.7 Estimated Speedup and Efficiency of PSGP4 for Various Hypercube Sizes

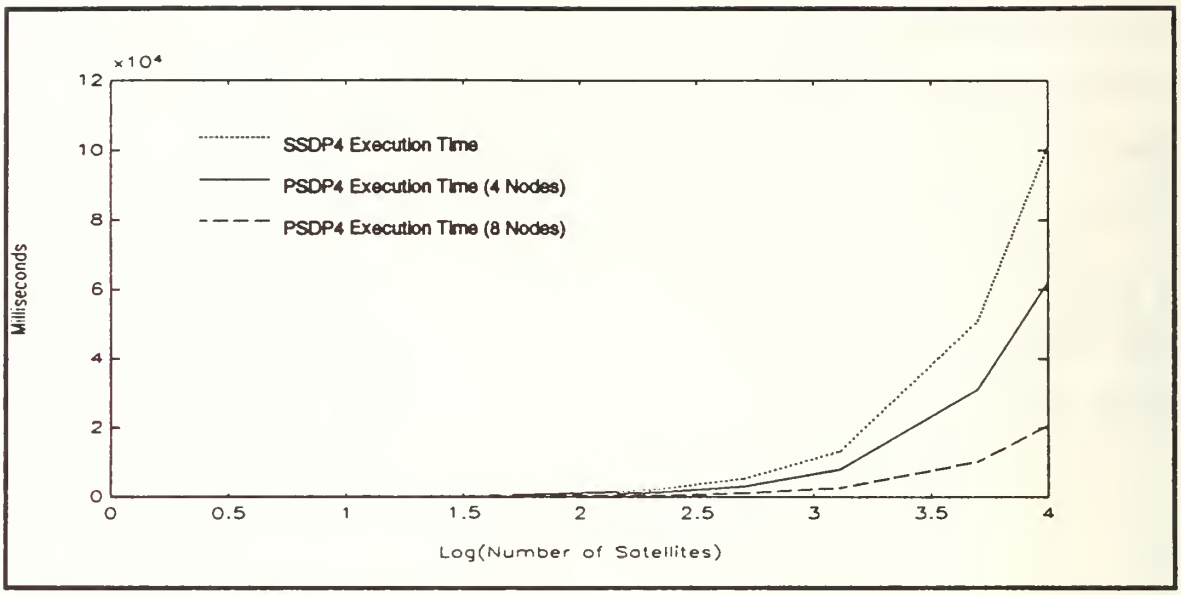

Figure 5.8 SDP4 Execution Times

TABLE 5.3 PSDP4 PERFORMANCE

Number of Satellites	Eight Nodes		Four Nodes	
	S_p	E_p	S_p	E_p
6	3.92	.490	1.70	.425
12	4.48	.560	1.71	.428
36	4.93	.616	1.73	.432
144	5.13	.641	1.73	.433
216	5.15	.644	1.73	.432
500	5.15	.644	1.71	.428
1296	5.14	.642	1.66	.416
5000	4.98	.623	1.64	.410
10000	4.91	.615	1.63	.408

* $t_M(1)$ = .0634 msec
* $t1$ = 10.8 msecs
* n = 1050 satellites

The value 1050 was chosen since approximately 15 percent of the 7000 satellites propagated a day at the AFSPACECOM are deep space. Figures 5.9 and 5.10 indicate the estimates of $t(p)$, S_p, and E_p using 4 to 1024 processors. Using this model PSDP4 is capable of achieving a maximum efficiency of about .9 using a 6 dimensional hypercube (64 nodes).

4. Comparing the Parallel Algorithms for PPT2, SGP, SGP4, and SDP4

Before comparing the four models, a few parameter values need to be presented from Phipp's assessment of PPT2. The time to propagate a single satellite, $t1$, using the parallel version of PPT2 is 11.2 milliseconds. The maximum efficiency achieved on an 8 node hypercube was .67. Using 4 nodes, the maximum efficiency was .45. The number of satellites propagated ranged from 12 to 20736. Using the same model presented in this thesis, derived by Phipps (1992), to predict program performance versus hypercube dimension with the following substitutions:

* $t_M(1)$ = .693 msec
* $t1$ = 11.2 msecs
* n = 1728 satellites,

the parallelized version of PPT2 is capable of achieving near .9 efficiency using a 4 dimensional hypercube (16 nodes).

One observation that can be obtained in comparing these four models, using the experimental data, is based on the execution time to propagate a single satellite, $t1$, and the resulting efficiencies for an eight-node and a four-node hypercube. Table 5.4 shows an increase in efficiency for both the eight-node and the four-node hypercube with an increase in satellite propagation time, $t1$. This conclusion is not surprising since an

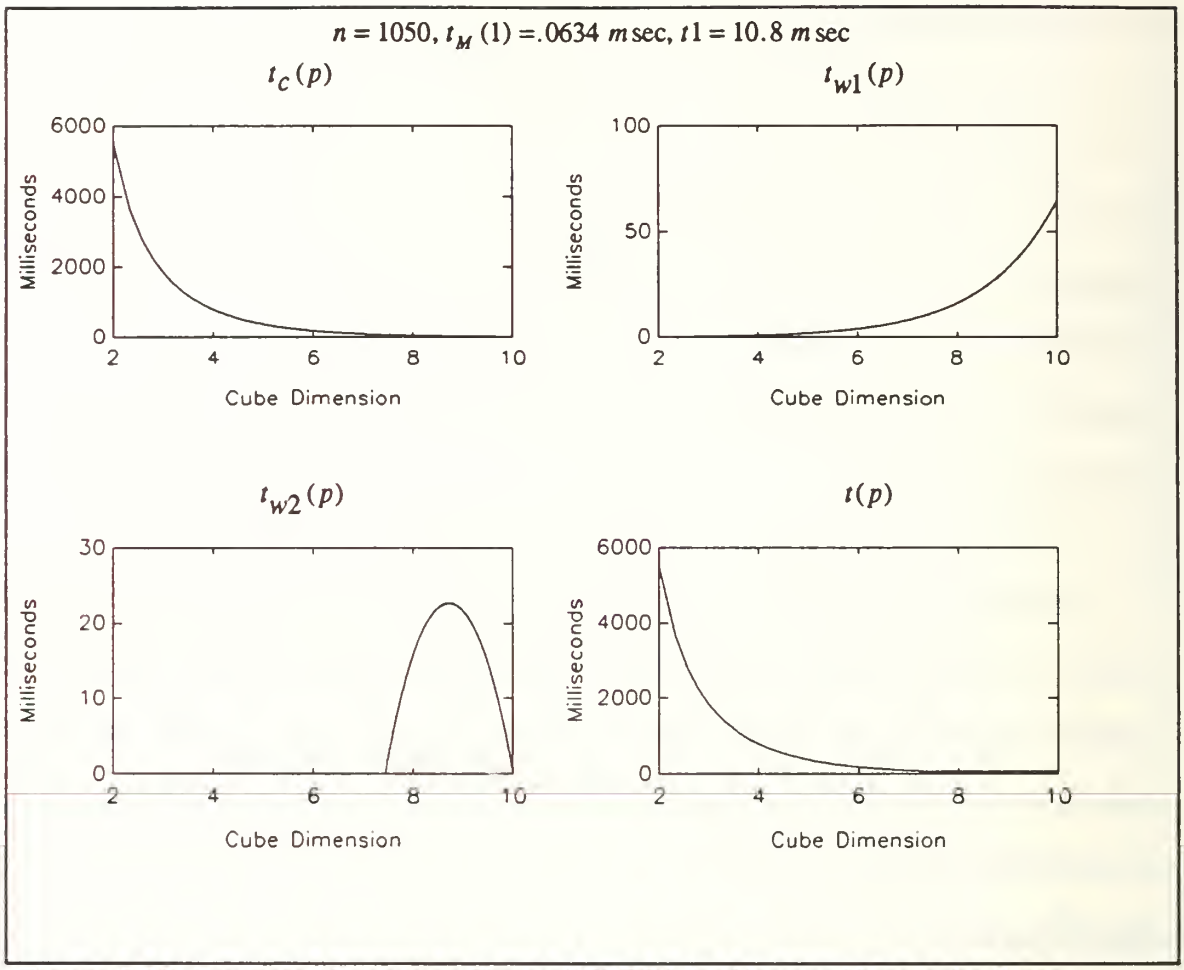

Figure 5.10 Estimated Execution Time of PSDP4 for Various Hypercube Sizes

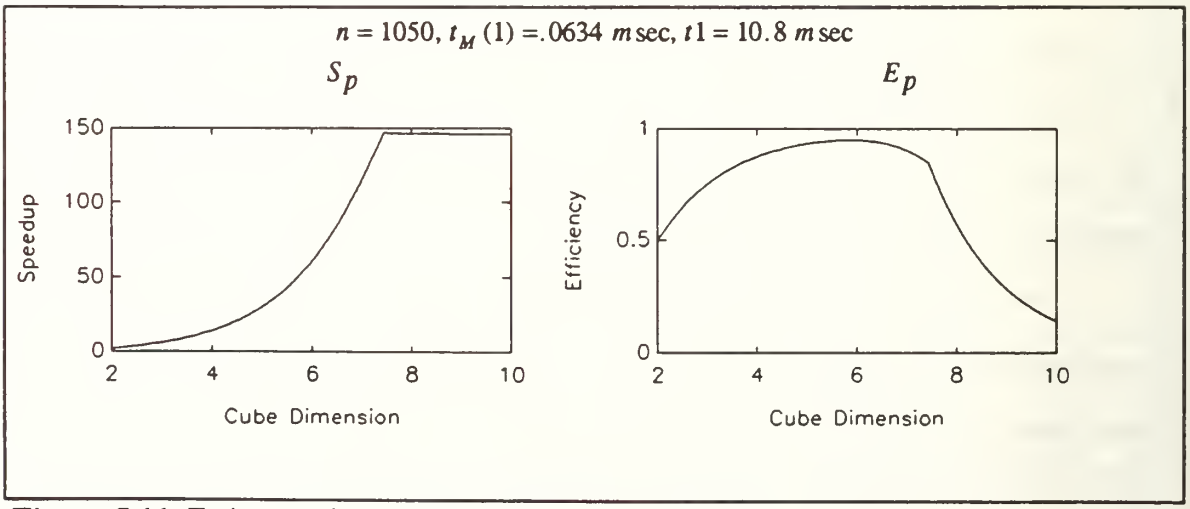

Figure 5.11 Estimated Speedup and Efficiency of PSDP4 for Various Hypercube Sizes

TABLE 5.4 PERFORMANCE COMPARISON

Satellite Motion Model	$t1$ (msecs)	Maximum E_8	Maximum E_4
SGP	4.6	.56	.38
SGP4	6.6	.60	.40
SDP4	10.8	.64	.43
PPT2	11.2	.67	.45

increase in execution time decreases the communication to computation ratio. This concept will be analyzed further in the next chapter.

VI. FURTHER ANALYSIS OF PHIPPS' DOMAIN DECOMPOSITION MODEL

The analysis in this chapter will only be concerned with SGP4 and SDP4 since these models are currently used operationally by the AFSPACECOM. Also, the resulting conclusions can be applied to SGP, since like SGP4 and SDP4, SGP is an analytic model with a comparable execution time. In the previous chapter Phipps' (1992) domain decomposition model was used to determine the optimal dimension hypercube for PSGP, PSGP4, and PSDP4. In this analysis the value of $t1$, the time to propagate a single satellite, represented only one call to the satellite motion subroutine resulting in one set of output for a specified time beyond epoch. This chapter will consider the effect of several calls to the subroutine per satellite, which is more realistic operationally. Each call corresponds to a specified time beyond epoch producing an output consisting of a position vector and velocity vector.

Another factor this chapter will address is the effects of the input node and output node (i.e., the reading and writing portions of the program) on the program performance. The model developed by Phipps (1992) to assess the performance of his domain decomposition strategy did not include timing the input node or output node. The iPSC/2 hypercube at the Naval Postgraduate School is not capable of concurrent reading or writing from the disk by its processors. In other words, only one node can access the disk at any given time. The hardware, however, does exist to give the hypercube this capability. Assuming his domain decomposition strategy will eventually be applied to a hypercube with this hardware or possibly another type of parallel machine with concurrent disk access capabilities, Phipps chose not to investigate the effects of the input node and output node at this phase of his research.

For the sake of completeness this chapter develops a model for Phipps' domain decomposition strategy using the iPSC/2 hypercube at the Naval Postgraduate School which includes the effects of the input and output nodes. The model's validity is obtained by comparing some of its theoretical program times with the corresponding experimental times. It is assumed the hypercube has its current capability where only one node can access the disk at any given time. The objective in analyzing this new model is to measure the performance of the **entire** domain decomposition strategy developed by Phipps (1992) when applied to the iPSC/2 hypercube at the Naval Post Graduate School, and use this assessment to further improve the parallelization strategy for SGP4, SDP4, and PPT2.

A. ASSESSMENT OF SEVERAL SUBROUTINE CALLS PER SATELLITE

Here Phipps' model will be used to determine the optimal dimension hypercube for SGP4 and SDP4 assuming a reasonable number of subroutine calls for each satellite motion model. The number of subroutine calls were chosen based on estimates received from the AFSPACECOM in Colorado Springs.

1. Assessment of SGP4

SGP4 propagates data for low earth satellites which require more frequent tracking than deep space satellites. Thus, a relatively large number of observations are received per day by the AFSPACECOM for each low earth satellite resulting in several calls to the SGP4 subroutine. Most likely the number of subroutine calls will fall between 50 and 100. An average value of 75 will be used in Phipps' model.

All parameter values will remain the same as presented in the assessment of SGP4 in the previous chapter except for $t1$, the time to propagate a single satellite. Each time a new set of satellite data is received by SGP4 an initialization subroutine is called before the SGP4 main subroutine is called (i.e., the subroutine that produces the position

and velocity vectors for the specified time since epoch). For every other incremented time specified for the same satellite, the initialization program is not called. Thus, the execution times for these specified times are slightly less than the computation time it takes for the first time value specified. Let these shorter execution times which are of equal value be represented by the parameter ts, and let tf denote the first execution time. Let m represent the number of subroutine calls per satellite. Therefore, $t1$ is expressed by

$$t1 = tf + (m-1)ts. \tag{6.1}$$

Thus, $t1$ is calculated to be

$$t1 = 6.6 + (74)(2.2) = 169.4 \ msec.$$

Figure 6.1 depicts the speedup and efficiency versus hypercube dimension obtained from Phipps' model and using the following substitutions:

* $t_M(1)$ = .0634 msecs
* $t1$ = 169.4 msecs
* n = 5950 satellites

Since the number of satellites, n, and the time to send a single message from the input node to the output node, $t_M(1)$, remains the same as for the m=1 case, a direct comparison can be made between the graphs in Figure 6.1 and Figure 5.7. Clearly, much higher speedups are obtainable for the m=75 case. If a large speedup is desired at the expense of efficiency, a maximum speedup near 1800 is achievable compared to a maximum speedup near 100 for the m=1 case. If maximum efficiency is desired the m=75 case depicts the potential to achieve near 100% efficiency using a hypercube of dimension eight (256 nodes) with a corresponding speedup near 250, compared to the m=1 case where the maximum efficiency is also nearly 100% but corresponding to a speedup just over 60 using a hypercube of dimension six (64 nodes). The above

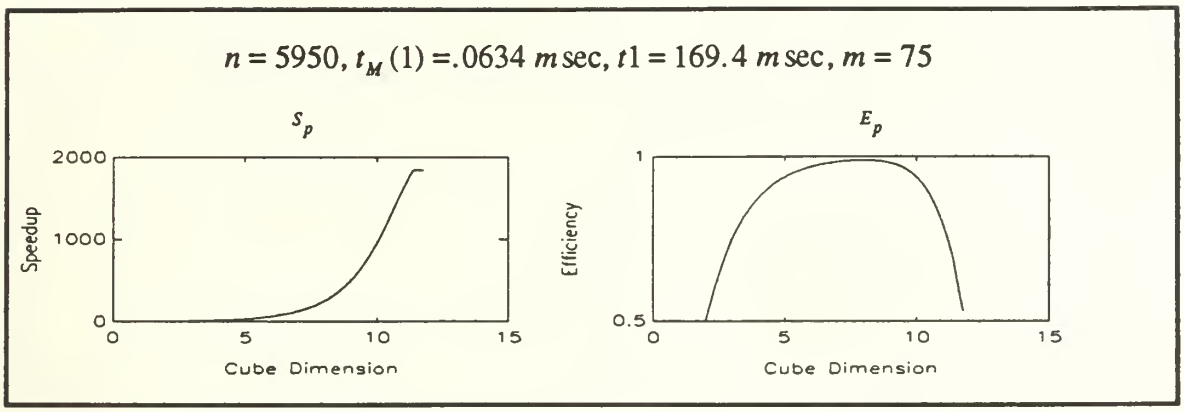

Figure 6.1 Estimated Speedup and Efficiency of PSGP4 for Various Hypercube Sizes

assessment is expected since a higher m yields an increase in the computation to communication ratio.

2. Assessment of SDP4

SDP4 propagates data for deep space satellites where the subroutine calls are likely to fall between 1 and 50. An average value of 25 will be used in Phipps' model. Thus, $t1$ is calculated to be

$$10.8 + (24)(4) = 106.8 \text{ msecs.}$$

Using the above value for $t1$ and the same substitutions for $t_M(1)$ and n as for the m=1 case,

* $t_M(1) = .0634$ msecs
* $n = 1050$ satellites,

Figure 6.2 depict the speedup and efficiency. In comparing Figure 6.2 with Figure 5.11 (the m=1 case), much higher speedups are observed with the m=25 case yielding a maximum speedup near 650 compared to a maximum speedup near 150 for the m=1 case. Considering maximum efficiency, the m=25 case has the potential to achieve near 100% efficiency using a hypercube of dimension seven (128 nodes) with a corresponding speedup of about 125. Whereas for the m=1 case the maximum efficiency is achieved with a hypercube of dimension six (64 nodes) and a corresponding speedup near 60. Again, these results are expected due to the increased computation time with a higher m.

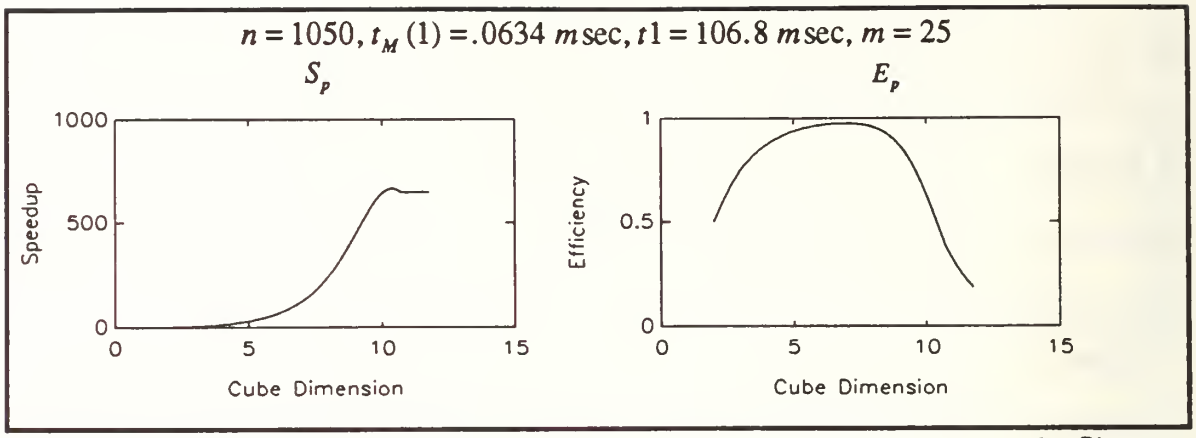

Figure 6.2 Estimated Speedup and Efficiency of PSDP4 for Various Hypercube Sizes

Notice, the speedup achievable is not as high as for PSGP4, because the number of subroutine calls per satellite is one-third that of PSGP4 (i.e., 25 calls compared to 75 calls).

B. REVISING PHIPPS' MODEL TO INCLUDE READS AND WRITES

Reading and writing to a disk is extremely costly in terms of computer time. As will be observed in assessing this revised model for SGP4 and SDP4, the initial access to the disk by the input node or output node produces a relatively large overhead for each of the programs. If the calculation portion of the satellite motion program isn't large enough in terms of computer time relative to the initial disk access time as well as the time to continue reading or writing from or to the disk after initial access, the parallel algorithm performance suffers greatly. This section of Chapter VI develops a model for Phipps' domain decomposition strategy which includes the effects of the input and output node, assuming the use of the iPSC/2 hypercube where concurrent disk access is unavailable.

1. Revised Model

Since the output node is always the last node to finish in Phipps' domain decomposition algorithm, the program total time, $t(p)$, can be determined by the sum of

the output node's total wait time, tw, and the total time to write to the disk, td. Therefore, the total time, $t(p)$, is represented by

$$t(p) = tw + td. \tag{6.2}$$

The total time to write to the disk, td, can be modeled by

$$td = td_i + (n)(m)(td_a) \tag{6.3}$$

where

td_i = initial disk access time which includes time to write labels for output columns

td_a = time to write data output (position and velocity vector) for each time specified (i.e., each time specified corresponds to a subroutine call)

n = number of satellites

m = number of subroutine calls per satellite.

The total wait time, tw, can be broken in two areas, the initial wait time, tw_i, the output node must wait before it can initially access the disk (i.e., assuming the input node (node zero) accesses the disk first which has been observed to be more likely), and the total sum of the remaining wait times, tw_a. The initial wait time, tw_i, is the total time for all the satellites to be read in by node zero. Thus, tw_i can be modeled by

$$tw_i = tr_i + (n-1)tr_a \tag{6.4}$$

where

tr_i = time for node zero to initially access the disk which includes reading the first satellite data set

tr_a = time to read each satellite data set after the first one.

The total sum of the remaining wait times, tw_a, depends on the time to propagate a single satellite, $t1$, by each of the working nodes and the number of working nodes. Since the initial disk access time, td_i, far exceeds $t1$ for SGP4 and SDP4, even with considering 75 subroutine calls for SGP4 and 25 subroutine calls for SDP4, there is no wait time by the output node between completing its initial disk access and writing its first set of output data. For all subsequent satellite data sets the wait time is also zero since $t1$, the time to

propagate the output for a single satellite, is found to be less than the time to write the satellite's output which may be modeled by

$$(m)(td_a) \tag{6.5}$$

for each of the satellite programs SGP4 and SDP4. This also implies using any more than two working processors will not improve program performance, therefore p, the number of processors, will equal four for this model.

Combining Equations 6.2 - 6.5 the total program time for PSGP, PSGP4, and PSDP4 can be modeled by

$$t(4) = tr_i + (n-1)tr_a + td_i + (n)(m)td_a. \tag{6.6}$$

Combining Equations 6.1 - 6.5, the time to execute SGP4 and SDP4 using one processor, $t(1)$, is expressed by

$$t(1) = tr_i + (n-1)tr_a + n[tf + (m-1)ts] + td_i + (n)(m)td_a. \tag{6.7}$$

Thus, using Equations 4.2 ,4.3, 6.6, and 6.7 the speedup and efficiency using four processors may be modeled by

$$S_4 = \frac{t(1)}{t(4)} = \frac{tr_i + (n-1)tr_a + n[tf + (m-1)ts] + td_i + (n)(m)td_a}{tr_i + (n-1)tr_a + td_i + (n)(m)td_a}$$

$$E_4 = \frac{S_4}{4} = \frac{tr_i + (n-1)tr_a + n[tf + (m-1)ts] + td_i + (n)(m)td_a}{4[tr_i + (n-1)tr_a + td_i + (n)(m)td_a]}. \tag{6.8}$$

During the development of this model, consideration was given to the scenario in which the input node reads only one satellite data set (or a small portion of the entire batch of satellite data sets), distribute to a working node (or working nodes) and then read the next satellite data set (or next portion of satellite data sets), then distribute, etc., vice reading the entire batch of satellite data sets and then distributing. Since the restriction still exists where only one of the input or output nodes can access the disk at

any given time while the other must wait, the results with this revised method of reading the disk would not present a significant difference in performance.

a. Assessment of SGP4

Table 6.1 gives values for speedup and efficiency for various n and m values (i.e., number of satellites and number of subroutine calls per satellite) using the revised model and the following substitutions obtained experimentally:

* tr_i = 514 msecs
* ts = 2.2 msecs
* tr_a = 18.2 msecs
* td_i = 542 msecs
* tf = 6.6 msecs
* td_a = 8.85 msecs

The single read and write times (tr_a and td_a) varied with the number of satellites and the number of subroutine calls per satellite (n and m). The read time per satellite increased somewhat as the number of satellites, n, increased (values ranging from 16.4 to 22.7 milliseconds), so an average of 18.2 milliseconds is used for tr_a. The write time for each subroutine call decreased somewhat as the number of subroutine calls per satellite, m, increased (values ranging from 12.1 down to 7.73 milliseconds), thus an average value of 8.85 milliseconds is used for td_a. Table 6.1 indicates minimal variance in speedup or efficiency with changes in m or n.

The speedup and efficiency were obtained experimentally using four and eight nodes for the m=5 case for various numbers of satellites, and are given in Table 6.2. These actual efficiencies and speedups indicate that the estimated performance values listed in Table 6.1 are low, but still reasonable. Table 6.2 reinforces the observation that more than four nodes does not improve performance, in fact the contrary is true.

The speedups and efficiencies depicted both theoretically and experimentally using four nodes are not bad considering there are only two working nodes. But,

TABLE 6.1 PSGP4 ESTIMATED PERFORMANCE

Number of Satellites (n)	Number of Subroutine Calls Per Satellite (m)									
	1		5		25		75		100	
	S_4	E_4	S_4	E_4	S_4	E_4	S_4	E_4	S_4	E_4
144	1.19	.298	1.22	.305	1.24	.310	1.25	.313	1.25	.313
500	1.23	.308	1.24	.310	1.25	.313	1.25	.313	1.25	.313
1050	1.24	.310	1.24	.310	1.25	.313	1.25	.313	1.25	.313
5950	1.24	.310	1.25	.313	1.25	.313	1.25	.313	1.25	.313

TABLE 6.2 PSGP4 ACTUAL PERFORMANCE

Number of Satellites (n)	Four Nodes		Eight Nodes	
	S_p	E_p	S_p	E_p
12	1.30	.325	1.03	.128
144	1.66	.411	1.59	.199
500	1.62	.406	1.61	.201
1296	1.60	.401	1.59	.199

obviously, this configuration is not desirable for SGP4 since the performance is limited to what can be achieved using four nodes.

b. Assessment of SDP4

Table 6.3 gives values for speedup and efficiency for various n and m values using the revised model with the following substitutions obtained experimentally:

* tr_i = 514 msecs
* ts = 4 msecs
* tr_a = 18.2 msecs
* td_i = 542 msecs
* tf = 10.8 msecs
* td_a = 8.85 msecs

Again minimal variance in speedup or efficiency is observed with changes in m or n.

The speedup and efficiency were obtained experimentally using four and eight nodes for the m=5 case for various numbers of satellites, and are given in Table 6.4. These actual efficiencies and speedups indicate again that the estimated performance values listed in Table 6.3 are low, but still reasonable. Table 6.4 also reinforces the observation that more than four nodes does not improve performance.

The performance of SDP4 yields better results than SGP4 since SDP4 has a higher propagation time per satellite, but this configuration for SDP4 is still limiting since it is restricted to four nodes.

c. Conclusion

Applying Phipps' domain decomposition strategy to SGP4, SDP4, and most certainly SGP (i.e., since SGP has an even smaller propagation time per satellite), on the iPSC/2 hypercube in the Mathematics Department at the Naval Postgraduate School which does not have concurrent disk access capability has limited performance capabilities (i.e., restricted to the use of four nodes). Most likely this conclusion will also hold for PPT2. But, there is great potential for Phipps' domain decomposition strategy if applied to a parallel computer that has the ability to perform concurrent disk access. The more processors that can access the disk simultaneously will likely result in more improved efficiencies and speedups.

TABLE 6.3 PSDP4 ESTIMATED PERFORMANCE

Number of Satellites (n)	Number of Subroutine Calls Per Satellite (m)									
	1		5		25		75		100	
	S_4	E_4	S_4	E_4	S_4	E_4	S_4	E_4	S_4	E_4
144	1.32	.330	1.38	.345	1.43	.358	1.45	.363	1.45	.363
500	1.37	.343	1.41	.353	1.44	.360	1.45	.363	1.45	.363
1050	1.39	.348	1.42	.355	1.44	.360	1.45	.363	1.45	.363
5950	1.40	.350	1.43	.358	1.45	.363	1.45	.363	1.45	.363

TABLE 6.4 PSDP4 ACTUAL PERFORMANCE

Number of Satellites (n)	Four Nodes ($m = 5$)		Eight Nodes ($m = 5$)	
	S_p	E_p	$\dot{S}_p$	E_p
12	1.32	.331	1.07	.134
144	1.80	.451	1.74	.218
500	1.74	.435	1.73	.216
1296	1.69	.423	1.69	.211

VII. CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

The overall objective of this thesis was to determine the parallel computing potential of the AFSPACECOM satellite motion models. The results given in Chapter V and Chapter VI indicate that Phipps' domain decomposition strategy holds promise, especially if applied to a hypercube with concurrent disk access capability. This algorithm is simple to apply. It provides the flexibility to vary the dimension of the hypercube and makes modifications to the satellite motion models easy to implement.

When omitting the effect of the input and output nodes, a maximum efficiency of only .56 was achieved for SGP, .60 for SGP4, and .64 for SDP4, but the potential efficiency is artificially bounded by the number of nodes available with the specific hypercube used. Having a maximum of only eight nodes available, which implies six working nodes, the efficiency of the domain decomposition algorithm is bounded above by .75. Using Phipps' domain decomposition model, it was shown that a maximum efficiency near 100% could be achieved for the m=1 case (i.e., one subroutine call per satellite) with a hypercube of 128 nodes for PSGP and a hypercube of 64 nodes for PSGP4 and PSDP4. In comparison, the parallelized version of PPT2 achieves a maximum efficiency at 16 nodes. In the case of higher m values (25 for PSDP4 and 75 for PSGP4) much higher speedups are obtainable yielding a maximum speedup of 1800 for PSGP4 and 650 for PSDP4, but at lower efficiencies. Also, these higher m values produced maximum efficiencies near 100% with a hypercube of 256 nodes for PSGP4 and a hypercube of 128 for PSDP4.

Including the effects of the input and output nodes significantly limits the performance capabilities of PSGP, PSGP4, and PSDP4 when applied to a hypercube without concurrent disk access capability. Here it was discovered that the performance is

restricted to what can be achieved using four nodes, which creates an upper bound of .5 for the efficiency corresponding to a speedup of two. Experimentally, a maximum efficiency of .41 was achieved with PSGP4 and .45 with PSDP4.

In conclusion, applying Phipps' domain decomposition strategy to a hypercube with concurrent disk access capability could result in speedup factors that would significantly reduce the time to predict state vectors for several thousand satellites.

The success in applying Phipps' domain decomposition strategy to SGP, SGP4, SDP4, and PPT2 using the iPSC/2 hypercube at the Naval Postgraduate School creates several possibilities for future research. One area of research would involve applying the parallelized versions of SGP, SGP4, SDP4, and PPT2 to higher dimension hypercubes that have concurrent disk access capabilities and validate the estimates presented by Phipps' domain decomposition model, thus omitting the effects of the input and output nodes. Then investigate the performance obtainable using the concurrent disk access capabilities which will be dependent upon the number of nodes that can access the disk simultaneously.

Another possible area of research would involve modifying the current satellite motion models to increase the accuracy of their predictions. The results in Chapters V and VI showed an increase in performance if the amount of computation was increased. Thus, greater accuracy could be achieved in far less time using Phipps' domain decomposition algorithm for SGP, SGP4, SDP4, and PPT2 than the time using the original serial algorithms. Also, from these results, applying this algorithm to satellite motion models that are more computationally intensive, such as semi-analytic models, would yield greater parallel computing potential.

Investigating other methods of parallelization presents another wide open area of research. One method of interest involves the use of Parallel Virtual Machines (PVM). This software provides a means to make a group of work stations act like a parallel

computer. Parallelizing through PVM is appealing, since work stations have already proven to be quite invaluable for the day to day tasks in business and academic organizations due to their versatility, affordability, and performance capabilities, and having the added ability of parallel computing opens up new areas of computer power. With the growing pace in parallel computer technology, research in applying satellite propagation models to the more recently developed parallel computers can lead to tremendous breakthroughs.

In conclusion, the results of Phipps' (1992) thesis combined with the results from this thesis clearly shows that satellite position prediction can be made more timely through parallel computing. Although the best method of parallelization may vary depending on the specific model used as well as the type of parallel computer used, parallel computing presents one sure avenue to achieve timely satellite position prediction for the growing number of earth orbiting objects.

APPENDIX A

INTEL iPSC/2 SPECIFICATIONS

This appendix contains a summary of the iPSC/2 hypercube multcomputer specifications as described in (iPSC/2 User's Guide, 1990, pp. 1-1 - 1-11). The exact performance values were obtained from (Arshi, 1988, pp. 17-22).

iPSC/2

<u>System Resource Manager (Host)</u>	
Central Processing Unit	INTEL 80386 (4 MIP)
Numeric Processing Unit	INTEL 80387 (250 KFLOP 64-bit)
Memory	8 Mbyte
External Communication	Ethernet TCP/IP local area network port
Operating System	AT&T Unix, Version V, Release 3.0
<u>Nodes</u>	
Node Processor	INTEL 80386 (4 MIP)
Numeric Co-processor	INTEL 80387 (250 KFLOP 64-bit)
Operating System	NX/2
<u>Internal Communication</u>	
Direct-Connect™ Routing	Message Latency -- 350 μsec Message Bandwidth -- 2.8 $\frac{Mbytes}{sec}$

APPENDIX B

SOURCE CODE LISTING

This appendix contains a listing of the host and node programs for the parallelized version of SGP, and SGP4/SDP4. Before each program set the input parameters are listed. Recall, SGP4 and SDP4 are two separate subroutines that are part of the same program. The host program for each satellite motion model loads the node program and clears the nodes once the process is complete. The node program contains the instructions for the nodes to complete their respective portions of the parallel algorithm. The node program assigns Node 0 to be the distributing node, the highest numbered node to be the collecting node, and the remaining nodes to be the working nodes. The working nodes execute the original SGP, or SGP4/SDP4 subroutine with only minor modifications.

SGP

Input Parameters

Parameter	Description
iept	End of file indicator
ts	Start time (first propagation time in minutes)
tf	Stop time (last propagation time in minutes)
delt	Time increment in minutes (determines number of propagation times)
epoch	Epoch reference time (year.day)
xndt2o	Time derivative of mean motion divided by two $\left(\frac{\mathrm{Re}\,\nu}{Day^2}\right)$
xndd6o	Second time derivative of mean motion divided by six $\left(\frac{\mathrm{Re}\,\nu}{Day^3}\right)$
xincl	Orbital inclination (degrees)

xnodeo	Right ascension of node (degrees)
eo	Eccentricity (degrees)
omegao	Argument of perigee (degrees)
xmo	Mean anomaly (degrees)
xno	Mean motion $\left(\frac{\text{Re}\,v}{Day}\right)$

There are a total of 13 parameters. Using double precison (i.e., 8 bytes per parameter), this forms a message size, from input node to a working node, of 104 bytes.

Host Program:

```
                    program PSGPh

*  This host program loads the node program PSGPn on the
*  nodes of the attached hypercube.  Upon completion of the
*  catalog of satellite data, the program clears the nodes for
*  another process.

*  Set host specific parameters

      data pid / 0 /

*  Set process id

      call setpid (pid)

*  Load program PSGPn on the nodes

      print *, 'loading nodes'
      call load ('psgpn', -1, pid)

*  Receive message that nodes are complete

      call crecv (99, istop, 4)
      print *, 'nodes complete'

*  Kill process on nodes

      call killcube (-1, -1)


      stop
      end
```

Node Program:

```
                    program PSGPn

*  This program propagates n-2 satellites concurrently, where n is
*  the number of nodes belonging to the attached cube.  Node 0 is
*  the distributing node and the Node n-1 is the collecting node.
*  The remaining nodes are the working nodes that propagate the
*  satellites using AFSPACECOM subroutine SGP.  For simplicity,
*  the tasks for all nodes are combined on this one node program.
*  Tasks are partitioned by logical if statements.

      implicit real * 8  (a - h, o - z)
      real * 8  sat (13, 25000), f (13), g (7 * 5 + 1)

      integer hostid, pid, outlen

      common / sg / f (13), g (7 * 5 + 1), ig

      data inlen / 104 /
      data isat / 1 /, n / 1 /, istop / 1 /, pid / 0 /

      maxlen = 8 * (7 * 5 + 1)

*** WARNING 5 IN THE ARRAY G AND IN THE VALUE MAXLEN IS THE MAXIMUM
*** NUMBER OF PROPAGATION TIMES FOR A GIVEN SET OF SATELLITE INPUT

      mynod = mynode( )
      numnode = numnodes( )
      hostid = myhost( )

*0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

*  Node 0 reads and distributes data among the working nodes

       if (mynod .eq. 0) then

*  Read complete catalog of satellite data

          open (30, file='satinp.l', status='old', form='formatted')
 10       read (30, *) (sat (j, isat), j = 1, 4)
          if (sat (1, isat)  .eq.  0.0) go  to  25
          read (30, 15) (sat (j, isat), j = 5, 10)
 15       FORMAT (F15.8, F10.8, E9.5, F8.4, F9.4, F9.7)
          read (30, 20) (sat (j, isat), j = 11, 13)
          if (sat (13, isat)  .le.  0.0) go  to  10
 20       FORMAT (F8.4, F9.4, F12.8)

          isat = isat + 1
          go  to  10
 25       close (unit = 30)
```

```
*   Send number of data sets to all nodes

      isat = isat - 1
      call csend (mynod, isat, 4, -1, pid)

*   Distribute satellite data sets to working nodes

      it1 = mclock( )
      iter = isat / (numnode - 2)
      do 1201  j = 1, iter
        do 1201  i = 1, numnode - 2
          call csend (mynod, sat (1, n), inlen, i, pid)
1201  n = n + 1
      iter = mod (isat, numnode - 2)
      if  (iter  .lt.  1) go  to  1203
      n = n - 1
      do 1202  j = 1, iter
1202  call csend (mynod, sat (1, n + j), inlen, j, pid)
1203  it2 = mclock ( )

*   End Node 0

*0000000000000000000000000000000000000000000000000000000000000000000000
*WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW

*   Begin working nodes

      else
        if (mynod .lt. numnode - 1) then

*   Receive number of data sets from Node 0 and compute total number
*   of satellites to propagate

          call crecv (0, isat, 4)
          it1 = mclock( )
          if (mynod .le. mod (isat, numnode - 2)) then
            iter = isat / (numnode - 2) + 1
          else
            iter = isat / (numnode - 2)
          endif

*   Receive satellite data, execute SGP, and send results to
*   collecting node

          do 1220  i = 1, iter
            call crecv (0, f, inlen)
            call sgpm( )
            outlen = 8 * ( 7* ig + 1)
1220      call csend (mynod, g, outlen, numnode - 1, 0)
          it2 = mclock( )
```

```
*  End Working Nodes

*WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
*CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

*  Begin Collecting Node

      else
        open (unit = 11, file = 'velpos2')
        write (11, 1235) 'Tsince', 'X', 'Y', 'Z', 'XDOT', 'YDOT', 'ZDOT'
1235    format (10x, a, 10x, a, 18x, a, 18x, a / 26x, a, 15x, a, 15x,a)

*  Receive total number of satellite sets

        call crecv (0, isat, 4)

*  Collect results and write to external file from Working Nodes

        it1 = mclock( )
        do 1240 k = 1, isat
           call crecv (-1, g, maxlen)
           ig = g(1)
          write (unit = 11, 1245) ((g(i), i = 2 + (j - 2) * 7, 8 + (j - 2) * 7), j = 2 , ig + 1)
1240    continue
        it2 = mclock( )
1245    format ( / 4f17.8 / 15x, 3f17.8)
        close (unit = 11)

*  Send message to Host that process is complete

        call csend (99, istop, 4, hostid, pid)

*  End Collecting Node

*CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      endif
      endif

      itot = it2 - it1
      print  *, 'node# ', mynod, ' time  ', itot, '  for ', isat, ' sats'

       stop
       end
```

Input Parameters

Here the input parameters are read in a little differently compared to SGP. The reading format is more compatible with the PC version of SGP4/SDP4 received from the AFSPACECOM. Also, this input is more complete in terms of object identification.

Parameter	Description
icrdno	Line number of data input
satn	Satellite number
yr	Year
rday	Day number
xndot	Time derivative of mean motion divided by two $\left(\frac{\mathrm{Re}\, v}{Day^2}\right)$
xn2dt	Second time derivative of mean motion divided by six $\left(\frac{\mathrm{Re}\, v}{Day^3}\right)$
ie	Exponent of xnd2dt
bterm	Modified ballistic coefficient (ER^{-1})
ie2	Exponent of bterm
ephtyp	Ephemeris type
icrdno2	Line number of data input
xinco	Orbital inclination (degrees)
xnodeo	Right ascension of node (degrees)
eo	Eccentricity (degrees)
omegao	Argument of perigee (degrees)
xmao	Mean anomaly (degrees)
xno	Mean motion $\left(\frac{\mathrm{Re}\, v}{Day}\right)$
iyr	Year of start time

srday	Day number of start time
jyr	Year of stop time
spday	Day number of stop time
delta	Time increment in minutes (Determines number of propagation times)

There are a total of 22 parameters. Using double precison this forms a message size, from input node to a working node, of 176 bytes.

Host Program:

```
                    program PSGP4h/PSDP4h

*  This host program loads the node program PSGP4n on the
*  nodes of the attached hypercube.  Upon completion of the
*  catalog of satellite data, the program clears the  nodes for
*  another process.

*  Set host specific parameters

      data pid / 0 /

*  Set process id

      call setpid (pid)

*  Load program PSGP4n on the nodes

      print *, 'loading nodes'
      call load ('psgp4n', -1, pid)

*  Receive message that nodes are complete

      call crecv (99, istop, 4)
      print *, 'nodes complete'

*  Kill process on nodes

      call killcube (-1, -1)


      stop
      end
```

Node Program:

```
                    program PSGP4n

*  This program propagates n-2 satellites concurrently, where n is
*  the number of nodes belonging to the attached cube.  Node 0 is
*  the distributing node and the Node n-1 is the collecting node.
*  The remaining nodes are the working nodes that propagate the
*  satellites using AFSPACECOM subroutine SGP4 for near-Earth
*  or SDP4 for deep-space objects.  For simplicity, the tasks for all
*  nodes are combined on this one node program.  Tasks are par-
*  titioned by logical if statements.

      implicit real * 8  (a - h, o - z)
      real * 8  sat (22), f (22), g (7 * 5 + 1)

      integer hostid, pid, outlen

      common / sg / f (22), g (7 * 5 + 1), ig

      data inlen / 176 /
      data  isat / 1 /, n /1 /, istop / 1 /, pid / 0 /
      maxlen = 8 * (7 * 5 + 1)

*** WARNING 5 IN THE ARRAY G AND IN THE VALUE MAXLEN IS THE MAXIMUM
*** NUMBER OF PROPAGATION TIMES FOR A GIVEN SET OF SATELLITE INPUT

      mynod = mynode( )
      numnode = numnodes( )
      hostid = myhost( )

*0000000000000000000000000000000000000000000000000000000000000000000000000000000

*  Node 0 reads and distributes data among the working nodes

       if (mynod .eq. 0) then

*  Read complete catalog of satellite data

          open (10, file='satinp4.l', status='old', form='formatted')
 10       read (10, *) (sat (j, isat), j = 1, 10)
          if (sat (1, isat)  .eq.  0.0) go  to  35
          read (10, *) (sat (j, isat), j = 11, 17)
          read (10, *) (sat (j, isat), j = 18, 22)
          isat = isat + 1
 35       close (unit = 10)


*   Send number of data sets to all nodes

        isat = isat - 1
        call csend (mynod, isat, 4, -1, pid)
```

```
*  Distribute satellite data sets to working nodes

      it1 = mclock( )
      iter = isat / (numnode - 2)
      do 1201  j = 1, iter
       do 1201  i = 1, numnode - 2
        call csend (mynod, sat (1, n), inlen, i, pid)
1201  n = n + 1
      iter = mod (isat, numnode - 2)
      if  (iter  .lt.  1) go  to  1203
      n = n - 1
      do 1202  j = 1, iter
1202  call csend (mynod, sat (1, n + j), inlen, j, pid)
1203  it2 = mclock ( )

*  End Node 0

*0000000000000000000000000000000000000000000000000000000000000000000000000000000
*WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW

*  Begin working nodes

      else
        if (mynod .lt. numnode - 1) then

*  Receive number of data sets from Node 0 and compute total number
*  of satellites to propagate

          call crecv (0, isat, 4)
          it1 = mclock( )
          if (mynod .le. mod (isat, numnode - 2)) then
            iter = isat / (numnode - 2) + 1
          else
            iter = isat / (numnode - 2)
          endif

*  Receive satellite data, execute SGP4/SDP4, and send results to
*  collecting node

         do 1220  i = 1, iter
           call crecv (0, f, inlen)
           call sgp4m( )
           outlen = 8 * ( 7* ig + 1)
1220     call csend (mynod, g, outlen, numnode - 1, 0)
         it2 = mclock( )

*  End Working Nodes

*WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
*CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
*  Begin Collecting Node

      else
        open (unit = 12, file = 'velpos4')
        write (11, 1235) 'Tsince', 'X', 'Y', 'Z', 'XDOT', 'YDOT', 'ZDOT'
1235    format (10x, a, 10x, a, 18x, a, 18x, a / 26x, a, 15x, a, 15x,a)

*  Receive total number of satellite sets

        call crecv (0, isat, 4)

*  Collect results and write to external file from Working Nodes

        it1 = mclock( )
        do 1240 k = 1, isat
           call crecv (-1, g, maxlen)
           ig = g(1)
          write (unit = 12, 1245) ((g(i), i = 2 + (j - 2) * 7, 8 + (j - 2) * 7), j = 2 , ig + 1)
1240    continue
        it2 = mclock( )
1245    format ( / 4f17.8 / 15x, 3f17.8)
        close (unit = 11)

*  Send message to Host that process is complete

        call csend (99, istop, 4, hostid, pid)

*  End Collecting Node

*CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      endif
      endif

      itot = it2 - it1
      print  *, 'node# ', mynod, ' time  ', itot, '  for ', isat, ' sats'

       stop
       end
```

APPENDIX C

EXECUTION TIMES

This appendix contains the experimental execution times for the parallel and serial versions of SGP, SGP4, and SDP4 for various numbers of satellites. The parallel execution times are for an eight-node and a four-node hypercube. There is only one subroutine call per satellite, and the time to read and write is not included (i.e., only the calculational portion is timed).

SGP

TABLE C.1 EXECUTION TIMES FOR PSGP AND SSGP

Number of Satellites	PSGP		SSGP (msec)
	8 Nodes (msec)	4 Nodes (msec)	
6	9.70	18.6	26.8
12	15.3	36.0	53.0
36	38.9	105	160
144	143	420	632
216	213	635	958
500	489	1470	2190
1296	1280	4090	5750
5000	4890	14700	21900
10000	9800	29500	43900

SGP4

TABLE C.2 EXECUTION TIMES FOR PSGP4 AND SSGP4

Number of Satellites	PSGP4		SSGP4 (msec)
	8 Nodes (msec)	4 Nodes (msec)	
6	11.4	24.3	37.7
12	19.4	47.4	74.7
36	50.3	140	224
144	190	560	894
216	283	843	1340
500	653	1980	3110
1296	1700	5340	8050
5000	6840	21000	31100
10000	14000	42400	62100

SDP4

TABLE C.3 EXECUTION TIMES FOR PSDP4 AND SSDP4

Number of Satellites	PSDP4		SSDP4 (msec)
	8 Nodes (msec)	4 Nodes (msec)	
6	15.7	36.2	61.5
12	27.3	71.3	122
36	74.2	212	366
144	285	845	1460
216	426	1270	2190
500	985	2970	5080
1296	2560	7920	13200
5000	10200	31000	50800
10000	20700	62200	102000

LIST OF REFERENCES

Amdahl, G. , "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities", *AFIPS Conference Proceedings*, v. 30, pp. 483-485, April 1967.

Arshi, S., et al, "Application Performance Improvements on the iPSC/2 Computer", *A Technical Summary of the iPSC/2 Concurrent Supercomputer*, INTEL Corporation, pp. 17-22, 1988.

Brouwer, D., "Solution of the Problem of Artificial Satellite Theory without Drag", *Astronomical Journal*, v. 64, pp. 378-397, 1959.

Brouwer, D. and Hori, G., "Appendix to Theoretical Evaluation of Atmospheric Drag Effects in the Motion of an Artificial Satellite", *Astronomical Journal*, v. 66, pp. 193-225, June 1961.

Danby, J. M. A., *Fundamentals of Celestial Mechanics*, p. 130, William Bell, Inc., 1989.

Fitzpatrick, P. M., *Principles of Celestial Mechanics*, Academic Press, New York, 1970.

Flynn, M. J., "Very High-speed Computing Systems", *Proceedings of the IEEE*, v. 54, pp. 1901-1909, December 1966.

Hilton, C. G. and Kuhlman, J. R., "Mathematical Models for the Space Defense Center", Philco-Ford Publication Number U-3871, pp. 17-28, November 1966.

Hoots, Jr., F. R., "The Relationship Between SGP4 Geopotential Terms and Brouwer Geopotential Terms", Office of Astrodynamics Applications HQ Fourteenth Aerospace Force, p. 9, January 1975.

Hoots, F. R., "Space Track Report No. 3", Aerospace Defense Command, December, 1980.

Hujsak, R. S., "A Restricted Four Body Solution for Resonating Satellites with an Oblate Earth", *AIAA* Paper Number 79-136, June 1979.

Hwang, Kai and Briggs, Faye A., *Computer Architecture and Parallel Processing*, pp. 6, 26, McGraw-Hill, Inc., 1984.

iPSC/2 User's Guide, INTEL Corporation, Order Number 311532-006, pp. 1-1 - 1-11, 4-1 - 4-6, June 1990.

iPSC/2 VAST User's Guide, INTEL Corporation, Order Number 311571-002, February, 1989.

Kozai, Y., "The Motion of a Close Earth Satellite", *Astronomical Journal,* v. 64, pp. 367-377, November 1959.

Lane, M. H., "The Development of an Artificial Satellite Theory Using a Power Law Atmosphere Density Representation", *AIAA* Paper Number 65-35, January 1965.

Lane, M. H. and Cranford, K. H., "An Improved Analytical Drag Theory for the Artificial Satellite Problem", *AIAA* Paper Number 69-925, August 1969.

Lyddane, R. H., "Small Eccentricities or Inclinations in the Brouwer Theory of the Artificial Satellite", *Astronomical Journal,* v. 68, pp. 555-558, 1963.

Office of Astrodynamic Applications HQ Fourteenth Aerospace Force, "An Approximate Formula to Obtain $\bar{a}_o$ From $\bar{n}_o$", CCZ Technical Note 74-15, pp. 1-6, December 1974.

Phipps, W. E., "Parallelization of the Navy Space Surveillance Center (NAVSPASUR) Satellite Motion Model", Thesis, Naval Postgraduate School, June 1992.

Phipps, W. E., Neta, B. and Danielson, D. A., "Parallelization of the Naval Space Surveillance Satellite Motion Model", *Journal Astronomical Sciences,* accepted for publication, 1993.

Quinn, M. J., *Designing Efficient Algorithms for Parallel Computers,* pp. 18-20, 25-30, 233-235, McGraw-Hill, Inc., 1987.

Schumacher, P., "Orbital Propagators in Current Use", NAVSPASUR, July 1991.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2. Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3. Chairman, Department of Mathematics Naval Postgraduate School Monterey, CA 93943	1
4. Commander U. S. Naval Space Command Dahlgren, VA 22448-5170	1
5. Professor B. Neta, Code MA/Nd Department of Mathematics Naval Postgraduate School Monterey, CA 93943	3
6. Professor D. Danielson, Code MA/Dd Department of Mathematics Naval Postgraduate School Monterey, CA 93943	3
7. Professor R. Panholzer, Code EC/Pz Chairman, Space Systems Academic Group Naval Postgraduate School Monterey, CA 93943	1
8. Dr. Shannon Coffey Mathematics and Computer Technologies Section Naval Research Laboratory Naval Center for Space Technology, Code 8242 Washington, DC 20375-5000	1

9. CPT Warren E. Phipps, Jr. — 1
   U. S. Military Academy
   Department of Mathematical Sciences
   West Point, NY 10996

10. Paul J. Cefola, Ph. D — 1
    Draper Laboratory
    Decison and Control Systems
    555 Technology Sq.,
    Cambridge, MA 02139-3563

11. Dr. Stephen H. Knowles — 1
    U. S. Naval Space Surveillance Center
    Analysis and Software Department
    Special Projects Division, Code 837
    Dahlgren, VA 22448

12. Dr. Paul Schumacher — 1
    U. S. Naval Space Surveillance Center
    Analysis and Software Department
    Special Projects Division, Code 837
    Dahlgren, VA 22448

13. Denise Kaya and Dan Snow — 2
    HQ AFSPACECOM/CNY
    150 Vandenberg St, Suite 1105
    Peterson AFB CO 80914-4110

14. Tim Payne and Bob Morris — 1
    HQ AFSPACECOM/CNA
    150 Vandenberg St, Suite 1105
    Peterson AFB CO 80914-4110

15. LTCOL Rabins — 1
    1CACS/CC
    1 Norad Road
    Suite 3105
    Cheyenne Mountain AF Base, CO 80914-6009

16. LT Sara R. Ostrom — 3
    14 Deene Ct.
    Stafford, Va 22554